# AN EFFICIENT ZETA-VARIABLE PROPER ORTHOGONAL DECOMPOSITION-BASED REDUCED-ORDER MODEL FOR COMPRESSIBLE FLOWS WITH AEROELASTIC APPLICATIONS

A Dissertation

by

ELIZABETH H. KRATH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

| | |
|---|---|
| Chair of Committee, | Paul Cizmas |
| Committee Members, | Helen Reed |
| | Antony Jameson |
| | Raytcho Lazarov |
| Head of Department, | Srinivas Vadali |

May 2021

Major Subject: Aerospace Engineering

# ABSTRACT

This dissertation presents an efficient proper orthogonal decomposition (POD) based reduced-order model (ROM) for use in compressible fluid flows. The method is made efficient by rewriting the governing equations of fluid dynamics using zeta-coordinates, *i.e.*, using primitive variables with specific volume substituted in for density. This substitution allowed for the pre computation of the inner products, or coefficients, of the reduced-order model system of equations, and reduced the nonlinearity of the system. To stabilize the POD-based ROM, three methods were used: (1) the penalty method, (2) artificial dissipation, and (3) a new method that modifies the number of modes used in the POD approximation. Energy-based stabilizing methods to calculate the parameters in the penalty method and the artificial dissipation method were developed to improve the robustness of these methods for predicting new flow solutions. Furthermore, a minimization method to calculate the parameter in the penalty method was also developed.

The POD-based ROM was used to predict solutions for several cases of varying degrees of unsteadiness: for an outlet pressure oscillation and for deforming boundaries. The results of these cases are shown for a quasi-one-dimensional nozzle, a two-dimensional channel, a three-dimensional axisymmetric nozzle, NASA Rotor 67, the 10th standard configuration, and the 11th standard configuration. A stability analysis was performed using Lyapunov's method, and showed that the stabilizing methods used were successful in stabilizing the POD-based ROM. To compare the speedup attained by the POD-based ROM, its CPU time was compared to the full CFD solver, or full-order model, and to a ROM that uses density instead of specific volume. The POD-based ROM attained a speedup of greater than four orders of magnitude when compared to both, and was able to accurately predict new flow solutions.

# DEDICATION

The author would like to dedicate this work to God for helping her understand how to model His creation, to her twin for being MSU, to her family for their continuous support, to Grandma Choi for always praising her “astronaut” degree, and to her Oasis family for making these years the best of her life.

# ACKNOWLEDGMENTS

When I first started research, it was out of a desire to complete my honors requirements. I was convinced since I started my undergraduate studies in 2012 that I would never go to graduate school. In the spring of 2015, I took AERO 351, Aerothermodynamic Propulsion, which was taught by my now advisor, Dr. Cizmas. Near the end of the semester, I was asked to research with him. This shocked me, because I never expected that professors would go out of their way to ask students to research with them. I am very grateful that he did. When I joined his group as an undergraduate, I became convinced to continue this research into graduate school.

I want to thank Dr. Cizmas for his guidance, patience, and excellent mentorship throughout my undergraduate and graduate studies. I'd also like to thank him for taking the first step to get students into research. I've learned a lot from him over these past five years. I would also like to extend my thanks to my committee members, Dr. Helen Reed, Dr. Antony Jameson, and Dr. Raytcho Lazarov. Thank you also to Dr. Hara for your input during my preliminary exam.

I would also like to thank Dr. Andrew Gyekenyesi of the Ohio Aerospace Institute (OAI) for funding a part of this dissertation via a subcontract to OAIs Versatile Advanced Affordable Turbine Engines (VAATE) III, FA8650-14-D-2410, issued by USAF/AFMC, AFRL Wright Research Site.

Furthermore, I'd like to thank the Office of Graduate and Professional Studies (OGAPS) for the Graduate Diversity Fellowship that helped fund the remaining portion of my graduate studies.

I would not have progressed as far as I have today without the help of my fellow graduate students. To Raymond Fontenot, thank you for being the best office buddy. It made the office more comfortable having you, Fran, and Gaston around. To Neil Matula, thank you for being the first to include me in the group, and for being willing to listen to my ideas and provide feedback in kind. To Forrest Carpenter, thank you for your help and for answering my questions when I got stuck running cases. To Michael Polewski, you were a suitable replacement for Raymond after he left; thank you for being willing to listen to my ideas. I'd also like to thank Brian Freno, who wasn't a graduate student at the time but is my predecessor, for being willing to answer all of my questions

from POD to life in general.

Finally, I would like to thank my family for their support. Thank you to my parents, Mike and Veronica, to my brothers, Owen, Jonathan, Matthew, and Pejman, and to my sisters, Becky, Mary, Christy, and Victoria. I would not be where I am today without y'all.

CONTRIBUTORS AND FUNDING SOURCES

## Contributors

This work was supported by a dissertation committee consisting of Professor Cizmas [advisor], Professor Jameson, and Professor Reed of the Department of Aerospace Engineering and Professor Lazarov of the Department of Mathematics. The author would also like to thank Dr. Hara of the Department of Aerospace Engineering for his input during her preliminary exam.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

## Funding Sources

The graduate study was supported by the Graduate Diversity Fellowship from Texas A&M University and through a subcontract to OAIs Versatile Advanced Affordable Turbine Engines (VAATE) III, FA8650-14-D-2410, issued by USAF/AFMC, AFRL Wright Research Site.

NOMENCLATURE

**Greek Letters**

| | |
|---|---|
| $\beta$ | Parameter of interest |
| $\gamma$ | Specific heat ratio |
| $\tilde{\gamma}$ | Gamma parameter |
| $\mathbf{\Gamma}$ | Tangent space to POD subspace |
| $\epsilon$ | Least-squares error |
| $\varepsilon$ | Relative error |
| $\zeta$ | Specific volume |
| $\kappa$ | Wavenumber |
| $\lambda$ | Eigenvalue |
| $\tilde{\lambda}$ | Operating points |
| $\tilde{\nu}$ | Artificial dissipation parameter |
| $\tilde{\boldsymbol{\nu}}$ | Artificial dissipation parameter diagonal matrix |
| $\rho$ | Density |
| $\tau$ | Penalty parameter |
| $\boldsymbol{\tau}$ | Penalty parameter diagonal matrix |
| $\bar{\bar{\boldsymbol{\tau}}}$ | Viscous stress tensor |
| $\phi$ | Basis function |
| $\varphi$ | Phase shift |
| $\Omega$ | Spatial domain |
| $\tilde{\omega}$ | Angular velocity |

| | |
|---|---|
| $\omega_x$ | Wheel speed of blade rotating about $x$-axis |

**Latin Letters**

| | |
|---|---|
| $a$ | Time coefficient |
| $A$ | Area |
| $\tilde{A}$ | Amplitude |
| $c$ | Speed of sound |
| $D$ | Dimension of domain |
| $E_i$ | Energy index |
| $E$ | Total energy |
| $\mathbf{F}$ | Prescribed boundary condition |
| $\mathcal{F}$ | Nonlinear functional |
| $\mathcal{G}$ | Grassmann manifold |
| $H$ | Total enthalpy |
| $J$ | Jacobian |
| $\mathcal{J}$ | Nonlinear functional |
| $\mathbf{K}$ | Boundary condition error |
| $m$ | Total number of modes |
| $M$ | Number of snapshots |
| $n$ | Number of modes |
| $N$ | Resolution of spatial domain |
| $N_R$ | Number of reference subspaces |
| $p$ | Pressure |
| $\mathbf{q}$ | State vector |
| $r$ | Radius |
| $\mathbf{R}$ | Autocorrelation matrix |

| | |
|---|---|
| $s$ | Entropy |
| $\mathbf{S}$ | Projection of Laplacian of basis functions |
| $\mathcal{S}$ | POD subspace |
| $t$ | Time |
| $T$ | Temperature |
| $u$ | $u$-Velocity |
| $v$ | $v$-Velocity |
| $\mathbf{v}$ | Velocity vector |
| $w$ | $w$-Velocity |
| $W$ | Relative velocity |
| $x$ | Spatial coordinate |
| $\mathbf{Z}$ | Zeta state vector |

**Mathematical Symbols**

| | |
|---|---|
| $(\cdot, \cdot)$ | Inner Product |
| $\langle \cdot, \cdot \rangle$ | Ensemble average |

**Subscripts**

| | |
|---|---|
| BC | Subset belonging to points along prescribed boundary |
| hub | Subset belonging to outlet hub boundary |
| out | Subset belonging to outlet boundary |

**Superscripts**

| | |
|---|---|
| $\hat{}$ | Approximate value |
| $*$ | Complex conjugate |
| $\tilde{}$ | Perturbation |
| $\bar{}$ | Spatial-averaged value |

**Acronyms**

| | |
|---|---|
| POD | Proper Orthogonal Decomposition |
| FOM | Full-Order Model |
| ROM | Reduced-Order Model |
| CFD | Computational Fluid Dynamics |
| PDE | Partial Differential Equation |
| ODE | Ordinary Differential Equation |

TABLE OF CONTENTS

Page

ABSTRACT ..... ii

DEDICATION ..... iii

ACKNOWLEDGMENTS ..... iv

CONTRIBUTORS AND FUNDING SOURCES ..... vi

NOMENCLATURE ..... vii

TABLE OF CONTENTS ..... xi

LIST OF FIGURES ..... xiv

LIST OF TABLES ..... xvii

1. INTRODUCTION ..... 1

    1.1 Background and Motivation ..... 1
    1.2 Literature Review ..... 2
    1.3 Current Contributions ..... 6
    1.4 Outline of Dissertation ..... 6

2. PROPER ORTHOGONAL DECOMPOSITION ..... 7

    2.1 Static Average and Static Basis ..... 7
    2.2 Dynamic Average and Dynamic Basis ..... 10

3. FLOW SOLVER ..... 12

    3.1 Physical Model ..... 12
        3.1.1 Rotating Reference Frame ..... 14
    3.2 Full-Order Model ..... 15
    3.3 Reduced-Order Model ..... 16

4. METHODOLOGY ..... 18

    4.1 Stability of the Reduced-Order Model ..... 18
        4.1.1 Lyapunov's Method ..... 18
        4.1.2 Penalty Method ..... 21
            4.1.2.1 Note on Lifting Method ..... 23

4.1.3 Artificial Dissipation ..... 24
4.1.4 Modified Number of Modes ..... 26
4.2 Basis Function Interpolation ..... 27
4.2.1 Enriching the Snapshot Database ..... 28
4.2.2 Grassmann Interpolation ..... 28
4.3 ODEPACK ..... 30

5. VERIFICATION AND VALIDATION OF ZETA-ROM ..... 31

5.1 Quasi-One-Dimensional Nozzle ..... 31
5.1.1 Effect of Lifting Method ..... 36
5.2 Channel Flow ..... 38
5.3 Axisymmetric Nozzle ..... 44
5.3.1 Oscillating Outlet Pressure ..... 46
5.3.2 Deforming Boundary ..... 50
5.4 Rotor 67 ..... 57
5.5 10th Standard Configuration ..... 64
5.6 11th Standard Configuration ..... 67

6. STABILITY ANALYSIS OF THE ZETA-ROM ..... 73

7. Computational Time Results ..... 76

7.1 Cost-benefit of POD-based ROM ..... 80

8. CONCLUSIONS ..... 82

8.1 Future Work ..... 83

APPENDIX A. ISENTROPIC BOUNDARY CONDITION FOR SPECIFIC VOLUME ..... 95

APPENDIX B. REDUCED-ORDER SPALART ALLMARAS EQUATIONS ..... 97

B.1 Reduced-Order Model ..... 98
B.1.1 Production, $\mathcal{P}$ ..... 100
B.1.2 Dissipation, $\mathcal{D}$ ..... 101
B.1.3 Eddy Viscosity, $\nu_t$ ..... 102

APPENDIX C. ZETA-VARIABLE POD-ROM USER'S MANUAL ..... 103

C.1 Software Installation ..... 103
C.1.1 `unsromBC` Installation ..... 103
C.1.2 `bfg` Installation ..... 105
C.2 Basis Function Generator (`bfg`) ..... 106
C.2.1 Running `bfg` ..... 106
C.2.2 `bfg` Input and Output Files ..... 108
C.2.2.1 `bfg` Input File ..... 108

C.2.2.2 `bfg` Output Files ..... 110
C.3 Interpolation for Off-Reference Conditions ..... 111
C.3.1 `grassman` Code ..... 111
C.3.1.1 `grass` Example ..... 113
C.4 Starting and Executing `unsromBC` ..... 113
C.4.1 Steps to start and execute ..... 114
C.4.2 Example Cases ..... 116
C.5 Input and Output Files for the $\zeta$-ROM ..... 118
C.5.1 $\zeta$-ROM Input Files ..... 118
C.5.1.1 Main input file ..... 118
C.5.1.2 Secondary input file: "const.dat" ..... 124
C.5.1.3 Auxiliary input files ..... 125
C.5.2 $\zeta$-Coordinate POD-ROM Output Files ..... 126
C.6 Input and Output Files for `convert` ..... 128
C.6.1 `convert` Input File ..... 128
C.6.1.1 Auxiliary Files ..... 129
C.6.2 `convert` Output File ..... 129
C.7 `splitout` and `combineout` ..... 130
C.7.1 `splitout` ..... 130
C.7.2 `combineout` ..... 131

LIST OF FIGURES

FIGURE Page


5.1 Effect of penalty-enforced boundary condition on ROM for a quasi-one-dimensional nozzle. ..... 33

5.2 Phase portrait of time coefficients for a quasi-one-dimensional nozzle..... 34

5.3 Comparison of the approximate time coefficients of the zeta-ROM versus the exact time coefficients for a quasi-one-dimensional nozzle with an outlet pressure oscillation with angular velocity $\tilde{\omega}$ = 1 rad/sec and amplitude $\tilde{A} = 0.02$..... 35

5.4 Off-reference ROM solution vs. FOM solution for a quasi-one-dimensional nozzle with an outlet pressure oscillation with angular velocity $\tilde{\omega}$ = 1 rad/sec and amplitude $\tilde{A} = 0.025$. The dashed lines are the ROM solution and the solid lines are the FOM solution..... 36

5.5 Off-reference ROM solution vs. FOM solution for a quasi-one-dimensional nozzle with an outlet pressure oscillation with angular velocity $\tilde{\omega}$ = 1.5 rad/sec and amplitude $\tilde{A} = 0.02$. The dashed lines are the ROM solution and the solid lines are the FOM solution. ..... 37

5.6 Results using the lifting approach on a quasi-one-dimensional nozzle with $\tilde{\omega} = 1$ rad/sec and $\tilde{A} = 0.02$..... 38

5.7 Computational mesh of the two-dimensional channel with 4257 nodes..... 39

5.8 Stabilizing effect of the penalty-enforced boundary conditions and modified number of modes on the ROM solution for a two-dimensional channel case with oscillating outlet pressure of angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$: (1) time coefficients derived from the FOM solution, (2) ROM solution, (3) ROM solution with penalty-enforced boundary conditions, (4) ROM solution with penalty-enforced boundary conditions and a modified number of modes..... 41

5.9 Time-averaged relative error between the ROM solution and the FOM solution for a two-dimensional channel flow with outlet pressure oscillation of angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$. ..... 42

5.10 Two-dimensional channel flow with an outlet pressure oscillation with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$: time variation of maximum relative error between the ROM and FOM solutions across all nodes. ..... 43

5.11 Computational mesh of the axisymmetric nozzle with 22,673 nodes. ..... 45

5.12 Time-averaged relative error between the ROM and FOM solutions for an axisymmetric nozzle with outlet pressure oscillation of angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$. ..... 47

5.13 Time-averaged relative error between the ROM off-reference solution and the FOM solution for an axisymmetric nozzle with outlet pressure oscillation with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.015$. ..... 48

5.14 Time-averaged relative error between the ROM off-reference solution and the FOM solution for an axisymmetric nozzle with outlet pressure oscillation with angular velocity $\tilde{\omega} = 15$ rad/sec and amplitude $\tilde{A} = 0.01$. ..... 49

5.15 Nozzle inlet deformation for amplitude $\tilde{A} = 0.2$. ..... 50

5.16 Time-averaged relative error between the ROM and FOM solutions for an axisymmetric nozzle deforming with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.2$. ..... 53

5.17 Time-averaged relative error between the ROM off-reference and FOM solutions for an axisymmetric nozzle with walls oscillating with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.3$. ..... 56

5.18 Time-averaged relative error between the ROM off-reference and FOM solutions for an axisymmetric nozzle with walls oscillating with angular velocity $\tilde{\omega} = 15$ rad/sec and amplitude $\tilde{A} = 0.2$. ..... 56

5.19 NASA Rotor 67: computational mesh with $N = 299,844$ grid nodes. ..... 57

5.20 Time-Averaged Relative Error of the ROM to the FOM for Rotor 67 with an outlet pressure oscillation with angular velocity $\tilde{\omega} = 38,059$ rad/sec and amplitude $\tilde{A} = 0.05$. ..... 61

5.21 NASA Rotor 67: time-averaged relative error between the off-reference ROM and FOM pressures for an outlet pressure oscillating with angular velocity $\tilde{\omega} = 38,059$ rad/sec and amplitude $\tilde{A} = 0.075$. A close-up of the error around the blade is shown at blade midspan while a close-up of the error around the shock is shown at around 95% blade span location measured from the hub. Note that the legend for the midspan contour plots is different from the other two plots. ..... 63

5.22 NASA Rotor 67: time-averaged relative error between the off-reference ROM and FOM state variables, for an outlet pressure oscillating with angular velocity $\tilde{\omega} = 37,765$ rad/sec and amplitude $\tilde{A} = 0.05$. ..... 64

5.23 10th standard configuration computational mesh ..... 65

5.24 10th standard configuration: location of maximum error for $x$-velocity for case 1. ... 66

5.25 11th standard configuration computational grid. ..... 68

5.26 Comparison of the isentropic Mach number of the zeta-variable ROM to UNS3D POD to the FOM. ..... 71

5.27 Comparison of the coefficient of pressure of the zeta-variable ROM to UNS3D POD to the FOM. The legend follows that of Figure 5.26..... 72

6.1 Effect of the penalty method and a modified number of modes on the eigenvalues of the zeta-ROM ODE system. The red markers denote the zeta-ROM without stabilizing methods imposed. The blue markers denote the zeta-ROM with stabilizing methods imposed. ..... 74

LIST OF TABLES

TABLE Page


5.1 Grid convergence of the quasi-one-dimensional nozzle. ..... 32

5.2 On- and off-reference cases for the quasi-one-dimensional nozzle with oscillating outlet pressure. ..... 32

5.3 Grid convergence of the two-dimensional channel. ..... 39

5.4 On-reference cases for the two-dimensional channel. ..... 40

5.5 Two-dimensional channel flow: maximum spatial and temporal relative error between the ROM and FOM, for all on-reference cases. ..... 44

5.6 Grids refinement of the axisymmetric nozzle. ..... 45

5.7 Grid convergence of the axisymmetric nozzle. ..... 45

5.8 On- and off-reference cases for the axisymmetric nozzle with an oscillating outlet pressure. ..... 46

5.9 Axisymmetric nozzle with outlet pressure oscillation: maximum spatial and temporal relative error between the ROM and FOM solutions, for all cases. ..... 49

5.10 On- and off-reference cases for the axisymmetric nozzle with an oscillating boundary deformation. ..... 51

5.11 Mean values of state variables, amplitudes and phase angles for a nozzle deformation of the axisymmetric nozzle for all on-reference cases. ..... 52

5.12 Axisymmetric nozzle with oscillating boundary deformation: exact and scaled mean values of state variables, amplitudes and phase angles for all off-reference cases. .... 55

5.13 Axisymmetric nozzle with oscillating walls: maximum spatial and temporal relative error between the ROM and FOM solutions. ..... 55

5.14 NASA Rotor 67: on- and off-reference cases for oscillating outlet pressure. ..... 58

5.15 NASA Rotor 67 with outlet pressure oscillation: mean values of state variables, amplitudes and phase angles. ..... 60

5.16 NASA Rotor 67 with outlet pressure oscillation: maximum spatial and temporal relative errors between the ROM and FOM solutions, for all on-reference cases. ..... 61

5.17 NASA Rotor 67: exact and scaled mean values of state variables, amplitudes and phase angles. ..... 62

5.18 NASA Rotor 67: maximum spatial and temporal relative error between the off-reference ROM and FOM state variables..... 63

5.19 10th standard configuration on- and off-reference cases ..... 65

5.20 Relative error of the ROM to the FOM for the 10th standard configuration..... 67

5.21 11th Standard Configuration Cases..... 69

5.22 Relative error of the ROM to the FOM for the 11th standard configuration plunging blades with $180^\circ$ inter blade phase angle..... 70

6.1 Effect of the penalty method and a modified number of modes on $\dot{V}(\mathbf{a})$..... 74

7.1 Breakdown of the CPU runtime of the FOM for three on-reference cases: (1) 2D channel with 7 modes, (2) 3D nozzle (P) with 9 modes, and (3), 3D nozzle (D) with 5 modes. ..... 77

7.2 CPU runtime comparison between the ROM and the FOM for all cases: (1) 2D channel with 7 modes, (2) 3D nozzle (P) with 9 modes, (3) 3D nozzle (D) with 5 modes, (4) Rotor 67 with 5 modes, and (5) the 10th standard configuration with 5 modes. ..... 77

7.3 Breakdown of CPU run time of ROM and comparison to theoretical upper limit of speedup. ..... 78

7.4 CPU runtime comparison of the zeta-variable ROM to a conservative-variable ROM for different solvers, CFL numbers, and sub-iterations for a 2D channel case with $N = 4257$ and $m = 6$. ..... 79

7.5 Computational runtime with and without additional post-processing of the zeta-variable POD ROM (zeta-ROM) versus UNS3D POD and the FOM for the 11th standard configuration plunging blades with $180^\circ$ inter blade phase angle..... 80

B.1 Constants of the S-A Equation ..... 99

C.12 Example of `ODE Modes` in `const.dat` file. ..... 125

# 1. INTRODUCTION

## 1.1 Background and Motivation

Advancements in computer hardware and infrastructure have opened up pathways toward high-fidelity model development of complex flow problems. Despite these advancements, the simulation of such flow problems remain computationally expensive. The high-fidelity model or full-order model (FOM) of these simulations is then limited in its computational speed by its numerical scheme and/or grid quality. This effectively limits the FOM in its use for design as well as for real-time feedback [1, 2].

To amend this expense, one can instead use model reduction procedures to create reduced-order models (ROM). These ROMs operate at a lower-fidelity than the full-order models. The model reduction procedures replace the larger system of partial differential equations (PDEs) of the full-order model by a much smaller system of ordinary differential equations (ODEs) that make up the reduced-order model. One such model reduction procedure is the Proper Orthogonal Decomposition (POD) method.

Proper Orthogonal Decomposition, otherwise referred to as Principal Component Analysis or the Karhunen-Loeve Decomposition, was developed by Karhunen and Loeve separately [3, 4] as a statistical method to extract optimal basis sets from an ensemble of observations. This optimal basis is found in such a way that the time-averaged approximation error between the POD approximation and the full-order model is minimized in a respective norm. Because the basis set is orthonormal, projecting the full-order model onto the basis creates the reduced-order model, where the ROM has significantly less degrees of freedom than the FOM.

When applying the POD method to the governing equations of fluid dynamics written in primitive or conservative form, a POD approximate appears in the denominator. This appearance has two effects. First, it increases the non-linearity of the system of equations. Second, it may require the recalculation of the coefficients of the ODEs that make up the ROM at every time step.

Both effects act as limitations to the computational speedup of the ROM.

Another limitation appears when POD is applied to aeroelastic simulations with moving meshes. The relationship between grid points in POD is index-based. Therefore, unless the mesh deforms in a topologically consistent manner, that is, the grid deforms such that the relationship between grid points does not change in time, then the dynamics of the mesh deformation will not be captured. Therefore, the accuracy of the ROM diminishes.

The research proposed herein corrects these limitations in speedup of the reduced-order model by rewriting the governing equations of fluid dynamics in zeta-coordinates, that is, as a function of specific volume instead of density [5, 6]. A dynamic average and dynamic basis are used in the POD approximation such that the ROM can accurately model flows with deforming meshes without interpolating or modifying the boundary conditions. The zeta-coordinate formulation reduces the nonlinearity of the governing equations and allows for the pre-computation of the coefficients of the ODEs. As such, the computational speedup of the ROM is increased.

### 1.2 Literature Review

Proper Orthogonal Decomposition has been used for many applications. In particular, the first application of POD in fluid flows was performed by Lumley on turbulent flows [7]. Since then, POD has been used for turbomachinery [8, 9], cavity [10], and multi-phase [11, 12] flows among others. Although referred to differently in other fields, all POD methods: principal component analysis, Karhunen-Loeve decomposition, and singular value decomposition are equivalent [13]. When taking that into account, POD has also been used in chemistry [14, 15], biology [16], and statistics [17].

Other model reduction procedures have also been explored. The Balanced POD developed by Rowley uses empirical Gramians to find balancing transformations for large systems to correct the limitations of balanced truncations [18]. Unfortunately, this method is limited to linear, stable systems, although it may be extended to nonlinear systems. Another method developed is the bi-orthogonal decomposition, which is a special case of POD, developed by Aubry [19]. This method decomposes its data set into orthogonal temporal modes and orthogonal spatial modes

simultaneously. Schmid developed the dynamic mode decomposition that calculates dynamic modes, which are orthogonalized in time rather than space [20]. The dynamic modes provide a way of identifying coherent structures particularly for experimental data sets. An extensive review of model reduction procedures was presented by Lucia *et al.*, which includes a review on Volterra theory, harmonic balance, and the traditional POD method [21]. Furthermore, Dowell *et al.* presented a review of reduced-order models in unsteady aerodynamics [22]. Of note in his review is that POD is not necessarily limited to linear problems. While the list of methods given here is not a comprehensive list, it is clear that these model reduction procedures have become an important part of CFD model development.

Extensions to the traditional POD method have also been investigated. One common modification is to manipulate the form of the governing equations, for example, by linearization. Another is to change the form of the inner product of the Galerkin projection procedure. For example, Barone changed the form of the inner product to a symmetry inner product that better reflected the flow dynamics [23]. One can also change the form of the POD approximation. Brenner augmented the POD approximation with discontinuity modes to better capture problems with moving discontinuities [12].

Modifications are needed to the traditional POD method to capture aeroelastic effects. Because POD uses index-based relationships between grid points, then any mesh deformation, unless topologically consistent, will not be communicated to the POD modes. There have been several methods developed to correct this limitation. Anttonen developed a multi-POD ROM that uses a set of POD solutions trained for different grid deformations [24, 25, 26]. The POD solutions are then selected that best match the current grid deformation. Lewin augmented the POD approximation with control modes that reflect the heaving, lagging, and angular velocity of an airfoil [27]. This method is limited to modes with homogenous boundary conditions. Freno addressed the mesh deformation directly by augmenting the POD approximation with dynamic averages and dynamic basis functions [28, 29]. These dynamic modes are no longer functions of just space only, but are also dependent on time-dependent parameters associated with the flow unsteadiness.

One drawback of the POD method is its lack of guarantee to preserve the stability of the FOM [30]. Although the original space of the FOM may be stable, the subspace of the ROM may see unstable trajectories. This limits the robustness of the ROM for handling changes in geometry and flow conditions. While this is the case, it does not mean that the ROM cannot be stabilized.

There have been several methods developed that can aid in regaining the stability lost through the POD process. For example, extensions on the traditional POD method for stability have been performed for linear-time invariant ROMs by modifying the reduced-order basis coming from the right eigenvector [31, 32]. Other research on linear-time invariant systems focused on eigenvalue reassignment of the ROM system matrix [33]. It is possible to extend this eigenvalue reassignment for linear-time invariant systems to nonlinear systems through an iterative approach developed by Tomas-Rodriguez [34]. Barone and Kalashnikova found that a stability-preserving inner product could be developed respective to the type of flow problem [23, 35]. Artificial dissipation has also been successfully used to stabilize the ROMs after the fact by adding an artificial dissipation term to the governing equations [36, 37, 38]. A method to calculate the artificial dissipation such that the system energy of the ROM would decrease was developed by the author for off-reference conditions [39, 40, 41].

It has been found that the lack of enforcement of boundary conditions in a ROM can lead to constrained solutions, which may lead to instabilities in the ROM [30]. Therefore, enforcing boundary conditions in a ROM is another way to preserve stability. The penalty method can be used for such a purpose. This method penalizes the governing system of equations based off of how well the system satisfies the boundary conditions. When using the penalty method for stability, the question becomes how to calculate the penalty parameters. Hesthaven developed a stable penalty method for the linearized compressible navier-stokes equations by calculating the penalty parameter such that the system eigenvalues are negative [42]. This method was then extended to reduced-order models by Kalshnikova [43]. Sirisup used a bifurcation tracking software, AUTO, to calculate the penalty parameters for autonomous systems [37, 44]. A method to apply the penalty method to nonautonomous, nonlinear systems was developed by the author where the penalty parameter is

calculated such that a prescribed boundary error was set to zero [41].

The main benefit of a ROM consists in its ability to efficiently predict flow conditions beyond what is predicted by the FOM, that is, at off-reference conditions. To predict off-reference conditions, the basis functions, which are functions of space (and sometimes time), need to be interpolated to the new flow conditions. This can be done in two ways. The first method is enriching the snapshot database with multiple FOM solutions of differing values of the parameter of interest [11]. The second is interpolation using the tangent space to a Grassmann manifold, which was developed by Amsallem for POD-based reduced-order models [45]. The second method is ideal for interpolation, because it retains the orthonormality constraint of the POD basis functions. When used for off-reference solutions, the second method produces more accurate results [46].

With the ability to predict off-reference solutions, the POD-based ROM can be used for optimization studies. This opens up many possibilities to real-time feedback, control, and/or design. The main questions to consider when using POD for optimization is how to maintain the accuracy of the off-reference solutions and how to preserve the efficiency of the POD process throughout the optimization.

The accuracy of the POD-based ROM solution for off-reference solutions has different factors associated to it such as the interpolation of the basis, the calculation of the time coefficients, and the size of the design space. For example, Fahl addressed the size of the design space by using a trust region algorithm to update the POD-based ROM with new basis functions from snapshots of the FOM [47]. My-Ha addressed the interpolation of the basis functions and calculation of the time coefficients by creating an ensemble of solution snapshots with differing parameters to develop the POD basis functions, and then using an inverse design optimization with linear interpolation of the time coefficients [48]. Toal addressed the size of the design space by combining two kriging-based response surface optimizations with a POD-based approximation for the minimization of drag-to-lift [49]. Along with Toal, POD has been used for multiple aerodynamic shape optimization studies [27, 50, 51].

## 1.3 Current Contributions

This paper presents the development of a zeta-variable Proper Orthogonal Decomposition based reduced-order model for compressible flows with aeroelastic applications. To overcome the limitations that arise due to writing the governing equations in primitive or conservative coordinates, zeta-variables will be used. Zeta-variables replace density with specific volume in the governing equations. This substitution reduces the nonlinearity of the equations and allows for the pre-computation of the ODE coefficients that make up the reduced-order model [5, 6, 41]. Therefore, the computational speedup of the ROM to the FOM is increased. The POD approximation will be augmented with dynamic basis functions. These dynamic basis functions introduce flow unsteadiness into the basis functions themselves, no longer rendering them only functions of space. By doing this, it allows the POD method to continue to use index-based relations between grid points while still communicating mesh deformation to the modes.

The zeta-variable POD-based ROM will be verified and validated for several geometries of differing unsteadiness. For the same geometries, the POD-based ROM will be used to generate off-reference solutions, solutions for which the POD basis functions are not derived from. The off-reference solutions will be validated against the FOM.

## 1.4 Outline of Dissertation

The following section presents the governing equations of the full-order model written using the zeta variables. The POD-based reduced-order model is subsequently described. Results are then presented for a quasi-one-dimensional nozzle, a two-dimensional channel flow, an axisymmetric nozzle, NASA Rotor 67, and the tenth and eleventh standard configuration. A stability analysis of the reduced-order model is then presented along with the shortcomings of the analysis tools available. Finally, the computational runtime of the reduced-order model is discussed and compared to other respective models.

# 2. PROPER ORTHOGONAL DECOMPOSITION

Proper Orthogonal Decomposition (POD) extracts an optimal basis set from an ensemble of observations [3, 4]. When used in conjunction with a Galerkin projection procedure, it can significantly reduce the number of degrees of freedom of a problem. Ordinarily, the basis set that is found varies spatially and not temporally, which may lead to issues when computing domains with moving meshes [29]. Two methods are presented here: the traditional Proper Orthogonal Decomposition method with a static basis set and average and a modification to the POD method with a dynamic basis set and/or average.

## 2.1 Static Average and Static Basis[1]

Consider a set of discrete snapshots of some scalar function $q(x, t_i), 1 \leq i \leq M$, where $M$ is the number of snapshots. This set of snapshots is assumed to form a linear, finite-dimensional Hilbert space $L^2$ on a spatial domain $\Omega$. The POD method approximates the perturbation of the scalar function, $\tilde{q} = q - \bar{q}$, as a linear combination of some time-dependent orthogonal time coefficients, $a$, and time-independent orthogonal basis functions, $\phi$,

$$\tilde{q}(x, t_i) = \sum_{j=1}^{n^q} a_j^q(t_i)\phi_j^q(x) \tag{2.1}$$

where $n^q$ is the number of terms or modes kept to approximate $\tilde{q}(x, t_i)$. Alternatively, the POD approximation can be applied to the scalar function $q$

$$q(x, t_i) = \sum_{j=0}^{n^q} a_j^q(t_i)\phi_j^q(x) \tag{2.2}$$

where $a_0^q(t_i) = 1$ and $\phi_0^q(x) = \bar{q}$. Take note that the basis functions, $\phi$, are time-independent. The reconstruction (2.1) is optimal in the sense that the time-averaged least-square error of the POD

[1]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

approximation

$$\epsilon_{n^q}^q = \left\langle \left\| \tilde{q}(x,t_i) - \sum_{k=1}^{n^q} a_k^q(t_i)\phi_k^q(x) \right\|^2 \right\rangle \tag{2.3}$$

is a minimum for any $n^q \leq M$ combinations of basis functions [52]. Here $\|\cdot\|$ denotes the $L^2$ norm, $\|f\| = (f,f)^{1/2}$, where $(\cdot,\cdot)$ denotes an inner product, and $\langle\cdot\rangle$ denotes an ensemble average, $\langle f\rangle = \frac{1}{T}\int_0^T f(x,t)dt$. This minimization reduces to an eigenvalue problem of the form [53, p. 89]

$$\int_\Omega \langle \tilde{q}(x)\tilde{q}^*(x)\rangle \, \phi(y)dy = \lambda\phi(y) \tag{2.4}$$

which is a homogenous Fredholm integral equation of the second kind [54]. Here, the optimal basis functions are the eigenfunctions of the integral eigenvalue equation. The kernel of (2.4) is the autocorrelation function $R(x,y) = \langle \tilde{q}(x)\tilde{q}^*(y)\rangle$ where $*$ denotes a complex conjugate. The integral equation in (2.4) is solved using a Singular Value Decomposition (SVD).

If instead one has a set of discrete vector-valued observations

$$\mathbf{q}(\mathbf{x},t_i) = [q(x_1,t_i), q(x_2,t_i), ..., q(x_N,t_i)]^T$$

where $N$ is the resolution of the spatial domain, the discrete ensemble average becomes $\langle f\rangle = \frac{1}{M}\sum_{i=1}^M f(x,t_i)$ and the autocorrelation function becomes a tensor product matrix

$$\mathbf{R} = \frac{1}{M}\sum_{i=1}^M \tilde{\mathbf{q}}(\mathbf{x},t_i)\tilde{\mathbf{q}}^T(\mathbf{y},t_i) \tag{2.5}$$

where $\mathbf{R}$ is a symmetric, positive semi-definite matrix which by definition has orthogonal eigenvectors [55]. The eigenvalue problem in discrete form is

$$\mathbf{R}\boldsymbol{\phi} = \boldsymbol{\lambda}\boldsymbol{\phi}. \tag{2.6}$$

For this discrete set of observations, the method of snapshots developed by Sirovich [56] is used to calculate the eigenvectors of $\mathbf{R}$. This is efficient when $M \ll N$, since it reduces a system of $N$

equations to one of $M$ equations. Details on the implementation of this method are given in [9, 52]. To understand how much energy is captured by the modes, an energy index

$$E_j = \frac{\lambda_j}{\sum_{i=1}^{M} \lambda_i},\ j \in [1, n^q] \tag{2.7}$$

is used. Generally, the number of modes are selected such that 99% of the cumulative energy is captured.

A Galerkin projection procedure is used to reduce the size of the system of equations. For example, consider a scalar variable $q(x,t)$. The POD approximation of the time-derivative of this variable is

$$\dot{q}(x,t) = \frac{\partial}{\partial t} \sum_{j=0}^{n^q} a_j^q(t)\phi_j^q(x) = \sum_{j=0}^{n^q} \dot{a}_j^q(t)\phi_j^q(x).$$

where $\dot{\aleph}$ denotes the time-derivative. The variable $\dot{q}(x,t)$ is then projected along an optimal eigenvector subspace

$$(\dot{q}(x,t), \phi_k^q(x)) = \left( \sum_{j=0}^{n^q} \dot{a}_j^q(t)\phi_j^q(x), \phi_k^q(x) \right) = \sum_{j=0}^{n^q} \dot{a}_j^q(t) \left( \phi_j^q(x), \phi_k^q(x) \right) = \dot{a}_k^q(t)$$

and is contracted due to the orthogonality of the basis functions. When two variables are multiplied together

$$(q_1(x,t)q_2(x,t), \phi_k^{q_1}) = \left( \sum_{i=0}^{n^{q_1}} \sum_{j=0}^{n^{q_2}} a_i^{q_1} a_j^{q_2} \phi_i^{q_1} \phi_j^{q_2}, \phi_k^{q_1} \right) = \sum_{i=0}^{n^{q_1}} \sum_{j=0}^{n^{q_2}} a_i^{q_1} a_j^{q_2} \left( \phi_i^{q_1} \phi_j^{q_2}, \phi_k^{q_1} \right)$$

an inner product coefficient results, $\left( \phi_i^{q_1} \phi_j^{q_2}, \phi_k^{q_1} \right)$, that can be precomputed before solving the ODEs.

When an approximated variable is in the denominator, the inner product coefficient of that variable becomes time-dependent. For example, consider the inverse of the scalar variable $q$

$$\frac{1}{q(x,t)} = \frac{1}{\sum_{j=0}^{n^q} a_j^q(t)\phi_j^q(x)}.$$

When projected along an optimal subspace, the term is unable to be contracted

$$\left(\frac{1}{q(x,t)}, \phi_k^q(x)\right) = \left(\frac{1}{\sum_{j=0}^{n^q} a_j^q(t)\phi_j^q}, \phi_k^q(x)\right) \tag{2.8}$$

and therefore is time-dependent due to the presence of $a_j^q(t)$ in the inner product. This means that at every time step, the variable will need to be reconstructed using Eq. (2.1), and then the inner product in Eq. (2.8) will need to be recomputed. This greatly increases the computational cost of the POD-based reduced-order model. It is imperative then that the governing equations do not have any division of a state variable.

The only remaining unknowns after solving for the optimal set of basis functions are the time coefficients. Because the basis functions are orthogonal, the time coefficients can be calculated as

$$a_k^q = \frac{(\tilde{q}, \phi_k^q)}{(\phi_k^q, \phi_k^q)}. \tag{2.9}$$

While (2.9) produces the “exact" time coefficients that best approximate $\tilde{q}$, these time coefficients cannot be used for predicting the solution at off-reference conditions, that is, for conditions different from those predicted by the full-order model. For off-reference conditions, the time coefficients need to be computed by solving the system of ordinary differential equations obtained by substituting the approximation (2.1) into the governing equations and then projecting the approximate governing equations along the basis functions as was described before.

### 2.2 Dynamic Average and Dynamic Basis

Dynamic average and dynamic basis functions were developed to expand the capability of the POD-based ROM to model meshes with deforming or moving boundaries [29]. Consider the same set of discrete vector-valued observations as before, $q(\mathbf{x}, t)$. The basis functions are approximated as

$$\phi_k^q(\mathbf{x}) = \tilde{\phi}_0^{k^q}(\mathbf{x}) + \sum_{i=0}^{d} \tilde{\gamma}_i \tilde{\phi}_i^{k^q}(\mathbf{x}) \tag{2.10}$$

where $\Gamma = \{\tilde{\gamma}_i\}_{i=1}^{d}$ are parameters associated with the flow unsteadiness and $d$ is the number of parameters used, typically two or less. The gamma parameters must be linearly independent of each other. For $k = 0$, Eq. (2.10) gives the dynamic average, while for $k > 0$, the dynamic basis is given. For conciseness, Eq. (2.10) is written as

$$\phi_k^q(\mathbf{x}) = \sum_{i=0}^{d} \tilde{\gamma}_i \tilde{\phi}_i^{k^q}(\mathbf{x}),\ \tilde{\gamma}_0 = 1. \tag{2.11}$$

In this form, if one is using a dynamic average but no dynamic basis, then for $k, i > 0, \tilde{\gamma}_i = 0$. If one is using a dynamic basis but no dynamic average, then for $k = 0, i > 0, \tilde{\gamma}_i = 0$. The discrete variable $q(\mathbf{x}, t)$ can then be approximated as

$$\begin{aligned} q(\mathbf{x}, t) &= \sum_{i=0}^{n^q} a_i^q(t)\phi_i^q(\mathbf{x}) \\ &= \sum_{i=0}^{n^q} a_i^q(t) \sum_{j=0}^{d} \tilde{\gamma}_j \tilde{\phi}_j^{i^q}(\mathbf{x}) \\ &= \sum_{j=0}^{d} \tilde{\gamma}_j \sum_{i=0}^{n^q} a_i^q(t)\tilde{\phi}_j^{i^q}(\mathbf{x}). \end{aligned}$$

The dynamic average and dynamic basis are solved using either an eigenvalue algorithm or the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno algorithm [28].

By observation, it was noted that using the dynamic average or dynamic basis required fewer modes to capture the cumulative energy (2.7) than using a static average or static basis.

# 3. FLOW SOLVER

This chapter discusses the models used to simulate the flow dynamics. The physical model consists of the governing equations of fluid flow. Two models are then used to solve the physical model: a full-order model (FOM) and a reduced-order model (ROM). The FOM solves the discretized governing equations while the ROM solves the projection of the governing equations. These models are described in the following sections.

## 3.1 Physical Model

The Navier-Stokes equations for an isothermal, adiabatic flow in the absence of external forces are

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} &= 0 \\ \frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} &= -\frac{\partial p}{\partial x_i} + \frac{\partial \overline{\overline{\tau}}_{ij}}{\partial x_j} \\ \frac{\partial \rho E}{\partial t} + \frac{\partial \rho H u_i}{\partial x_i} &= \frac{\partial u_i \overline{\overline{\tau}}_{ij}}{\partial x_j} - \frac{\partial p u_i}{\partial x_i} \end{aligned} \tag{3.1}$$

where $\rho$ is the density, $u_i$ are the velocity components $(u, v, w)^T$, $x_i$ are the spatial components $(x, y, z)^T$, $p$ is the static pressure, $\overline{\overline{\tau}}_{ij}$ are the viscous stress tensor components, $E$ is the total energy, and $H$ is the total enthalpy. The viscous stress tensor for a Newtonian fluid is

$$\overline{\overline{\tau}}_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda \frac{\partial u_k}{\partial x_k} \delta_{ij}$$

where $\lambda$ is the bulk viscosity that can be defined in terms of the dynamic viscosity coefficient, $\mu$, using Stokes' hypothesis

$$\lambda = -\frac{2}{3}\mu. \tag{3.2}$$

Therefore, the viscous stress tensor becomes

$$\overline{\overline{\tau}}_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right). \tag{3.3}$$

The dynamic viscosity coefficient can be found using Sutherland's law

$$\mu = \frac{CT^{3/2}}{T+S} \tag{3.4}$$

where $C = 1.458 \times 10^{-6}\ \frac{\text{kg}}{\text{ms}\sqrt{\text{K}}}$ and $S = 110.4$ K.

Equation (3.1) can be expanded further by substituting in the definitions for the total energy and total enthalpy.

$$E = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{u_i u_i}{2}$$

$$H = h + \frac{u_i u_i}{2}$$

These substitutions along with assuming a perfect gas, $p = \rho RT$, give a form of the governing equations where the conservation of energy equation is now primarily written in terms of $p$. In index notation, these are

$$\frac{\partial \rho}{\partial t} + \rho\frac{\partial u_i}{\partial x_i} + u_i\frac{\partial \rho}{\partial x_i} = 0 \tag{3.5a}$$

$$\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j} + \frac{1}{\rho}\frac{\partial p}{\partial x_i} - \frac{1}{\rho Re}\left(\frac{\partial^2 u_i}{\partial x_i^2} + \frac{1}{3}\frac{\partial^2 u_j}{\partial x_i \partial x_j}\right) = 0 \tag{3.5b}$$

$$\frac{\partial p}{\partial t} + u_i\frac{\partial p}{\partial x_i} + \gamma p\frac{\partial u_i}{\partial x_i} - \frac{(\gamma-1)\mu}{Re}\frac{\partial u_i}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) = 0 \tag{3.5c}$$

where density appears in the denominator in the conservation of momentum equations. To remove the appearance of density in the denominator, specific volume, $\zeta$, can be substituted in for density. The equations then become

$$\frac{\partial \zeta}{\partial t} - \zeta\frac{\partial u_i}{\partial x_i} + u_i\frac{\partial \zeta}{\partial x_i} = 0 \tag{3.6a}$$

$$\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j} + \zeta\frac{\partial p}{\partial x_i} - \frac{\zeta}{Re}\left(\frac{\partial^2 u_i}{\partial x_i^2} + \frac{1}{3}\frac{\partial^2 u_j}{\partial x_i \partial x_j}\right) = 0 \tag{3.6b}$$

$$\frac{\partial p}{\partial t} + u_i\frac{\partial p}{\partial x_i} + \gamma p\frac{\partial u_i}{\partial x_i} - \frac{(\gamma-1)\mu}{Re}\frac{\partial u_i}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) = 0 \tag{3.6c}$$

where an appearance of a state variable in the denominator no longer occurs. The zeta-variable state

vector is

$$\mathbf{Z} = [\zeta \; u \; v \; w \; p]^T .$$

### 3.1.1 Rotating Reference Frame

The Navier-Stokes equations for a rotating reference frame about the $x$-axis written in zeta-variables are

$$\frac{\partial \zeta}{\partial t} - \zeta \frac{\partial u_i}{\partial x_i} + u_i \frac{\partial \zeta}{\partial x_i} - \zeta \omega_x \frac{\partial \tilde{x}_i}{\partial x_i} + \tilde{x}_i \omega_x \frac{\partial \zeta}{\partial x_i} = 0 \tag{3.7a}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} + \omega_x \tilde{x}_j \frac{\partial u_i}{\partial x_j} + \zeta \frac{\partial p}{\partial x_i} - \zeta \overline{\overline{\tau}}_{ij,j} = 0 \tag{3.7b}$$

$$\frac{\partial p}{\partial t} + u_i \frac{\partial p}{\partial x_i} + \gamma p \frac{\partial u_i}{\partial x_i} + \omega_x \tilde{x}_i \frac{\partial p}{\partial x_i} + \omega_x p \frac{\partial \tilde{x}_i}{\partial x_i} - (\gamma - 1) \overline{\overline{\tau}}_{ij} \frac{\partial u_j}{\partial x_i} = 0 \tag{3.7c}$$

where $\tilde{\mathbf{x}} = [0, z, -y]^T$ and $\overline{\overline{\boldsymbol{\tau}}}$ is the viscous stress tensor.

$$\overline{\overline{\boldsymbol{\tau}}} = \begin{bmatrix} \overline{\overline{\tau}}_{xx} & \overline{\overline{\tau}}_{xy} & \overline{\overline{\tau}}_{xz} \\ \overline{\overline{\tau}}_{xy} & \overline{\overline{\tau}}_{yy} & \overline{\overline{\tau}}_{yz} \\ \overline{\overline{\tau}}_{xz} & \overline{\overline{\tau}}_{yz} & \overline{\overline{\tau}}_{zz} \end{bmatrix}$$

The components of the viscous stress tensor in a rotating reference frame are

$$\begin{aligned}
\overline{\overline{\tau}}_{xx} &= \mu \frac{2}{3} \left( 2 \frac{\partial W_1}{\partial x} - \frac{\partial W_2}{\partial y} - \frac{\partial W_3}{\partial z} \right) \\
\overline{\overline{\tau}}_{yy} &= \mu \frac{2}{3} \left( 2 \frac{\partial W_2}{\partial y} - \frac{\partial W_1}{\partial x} - \frac{\partial W_3}{\partial z} \right) \\
\overline{\overline{\tau}}_{zz} &= \mu \frac{2}{3} \left( 2 \frac{\partial W_3}{\partial z} - \frac{\partial W_1}{\partial x} - \frac{\partial W_2}{\partial y} \right) \\
\overline{\overline{\tau}}_{xy} &= \mu \left( \frac{\partial W_1}{\partial y} + \frac{\partial W_2}{\partial x} \right) \\
\overline{\overline{\tau}}_{xz} &= \mu \left( \frac{\partial W_1}{\partial z} + \frac{\partial W_3}{\partial x} \right) \\
\overline{\overline{\tau}}_{yz} &= \mu \left( \frac{\partial W_2}{\partial z} + \frac{\partial W_3}{\partial y} \right)
\end{aligned}$$

where $W_i$ are the components of the relative velocity vector.

$$\mathbf{W} = \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 + z\omega_x \\ u_3 - y\omega_x \end{bmatrix}$$

### 3.2 Full-Order Model[1]

Two full-order models were used for generating the results presented herein. The first FOM solved the one-dimensional Euler equations written using characteristic variables [57]

$$\begin{gathered} \frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = 0 \\ \frac{\partial}{\partial t}\left(u + \frac{2c}{\gamma - 1}\right) + (u + c)\frac{\partial}{\partial x}\left(u + \frac{2c}{\gamma - 1}\right) + cu\frac{1}{A}\frac{\partial A}{\partial x} = 0 \\ \frac{\partial}{\partial t}\left(u - \frac{2c}{\gamma - 1}\right) + (u - c)\frac{\partial}{\partial x}\left(u - \frac{2c}{\gamma - 1}\right) - cu\frac{1}{A}\frac{\partial A}{\partial x} = 0 \end{gathered} \tag{3.8}$$

where $s$ is the entropy and $c$ is the speed of sound. MacCormack's technique was used to solve the FOM system [58]. Herein, this FOM is referred to as the 1D FOM.

The second full-order model solves the mass, momentum and energy conservation equations written using primitive variables, as opposed to the zeta variables used for the ROM. The governing equations of this FOM were solved using an in-house three-dimensional unstructured solver, called UNS3D [59]. This code solves both the Euler equations and the Reynolds-averaged Navier–Stokes equations using the finite volume method with Roe-Riemann flux splitting [60] and Harten entropy fix [61]. The code uses a dual-mesh cell-vertex discretization and an edge-based method [62]. An upwind method is used to calculate the convective fluxes. The gradients were computed using a weighted least squares with QR decomposition. The time integration can be done explicitly using a multi-stage Runge-Kutta method with local time stepping and residual smoothing, or implicitly, using a lower-upper symmetric Gauss-Seidel scheme [63]. The code was parallelized using Message

[1] A portion of this section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

Passing Interface and has deforming cell capabilities that satisfy the geometric conservation law. The FOM is second-order in time and space [64].

The UNS3D software is written in Fortran2003 and uses the MPI library for parallelization. The code was compiled and tested with several Fortran compilers: Absoft, Intel, and Gfortran, and has been tested on Unix and Linux operating systems. The UNS3D software is also able to produce POD-based reduced-order model results [46, 65]. In the scope of this dissertation, this ROM, referred to as UNS3D POD, will be used for comparison purposes.

### 3.3 Reduced-Order Model[2]

The snapshots generated by solving the FOM were used to assemble the **R** autocorrelation matrix (2.5). The POD basis functions $\phi$ were obtained as the eigenmodes of **R**. In the governing equations, (3.6) and (3.7), the state variables were approximated using (2.1) which yielded

$$\sum_{i=0}^{n^{Z_k}} \dot{a}_i^{Z_k} \phi_i^{Z_k}(\mathbf{x}) = \mathcal{J}_k\left(\mathbf{a}, \boldsymbol{\phi}\right), \quad k \in [1, D+2] \tag{3.9}$$

where $D$ is the dimension of the domain and $\mathcal{J}$ is a nonlinear functional consisting of the combinations of the time coefficients and basis functions of each approximated state variable. The approximated governing equations were projected along the POD basis functions, $\phi_j^{Z_k}, j \in [1, n^{Z_k}], k \in [1, D+2]$. Since the governing equations are nonlinear, the reduced system of equations is nonlinear as well. Projecting (3.9) along the POD basis functions contracts the time-derivative term

$$\dot{a}_j^{Z_k} = \left(\mathcal{J}_k\left(\mathbf{a}, \boldsymbol{\phi}\right), \phi_j^{Z_k}(\mathbf{x})\right), \quad j \in [1, n^{Z_k}],\ k \in [1, D+2]$$

such that the system of $N$ PDEs is reduced to a system of $m = \sum_{k=1}^{D+2} n^{Z_k}$ ODEs where the time coefficients are the dependent variables. The coefficients of the ODEs that appear in this equation are only dependent on space, and so can be precomputed before solving the system of ODEs. The

[2]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

system of ODEs can be concisely written as

$$\dot{\mathbf{a}} = \boldsymbol{\mathcal{F}}(\mathbf{a}) \tag{3.10}$$

where $\mathbf{a} \in \mathbb{R}^m$ and $\boldsymbol{\mathcal{F}} \in \mathbb{R}^m$. The inner product selected for this operation is the $L^2$ inner product, which when used with the velocity components yields the rate of kinetic energy

$$(\dot{\mathbf{v}}, \mathbf{v}) = \frac{1}{2}\frac{\partial}{\partial t}\left\|\mathbf{v}\right\|^2.$$

This is desirable, because the $L^2$ inner product reflects the physics of the flow [23]. The discrete $L^2$ inner product was used to project the governing equations

$$(f, g) = \int_\Omega f g d\Omega \approx \sum_{i=1}^{N} f(x_i) g(x_i) \Omega_i = \sum_{i=1}^{N} f(x_i) g(x_i)$$

where $\Omega \in \mathbb{R}^D$ is a reference spatial domain with unit length, width, and depth.

With the method presented here, the ROM was not developed by projecting the discretized governing equations of the FOM as was done in [11, 66], but was developed by projecting the differential form of the governing equations themselves. This may lead to small differences between the FOM and the ROM [67]. Furthermore, the ROM developed herein is not guaranteed to preserve the stability of the FOM. The stabilization of the ROM is discussed in the following section.

# 4. METHODOLOGY

This section discusses the methodology used in the zeta-variable POD ROM. Firstly, the stability of the reduced-order model is discussed followed by methods to stabilize the zeta-variable POD ROM. Secondly, methods to interpolate the basis functions to new flow solutions are discussed. Finally, a discussion on the ODE solver used to solve Eq. (3.10) is given.

## 4.1 Stability of the Reduced-Order Model

POD-based reduced-order models do not necessarily retain the stability of the FOM. This is due in part to the projection of the FOM onto the POD subspace. Although the FOM may be unconditionally stable, the POD subspace may end up seeing unstable trajectories [30]. Therefore, the following methods are considered for obtaining a stable ROM: (1) the penalty method, (2) artificial dissipation, and (3) a method that modifies the number of modes of the POD approximation within the governing equations.

Before explaining the methods used to stabilize the ROM, a discussion on how to determine the stability of an ODE system is given.

### 4.1.1 Lyapunov's Method

Lyapunov's method is a popular method to examine the stability of ODE systems [68]. For an indepth explanation of Lyapunov's method, see Slotine's book [69, Ch. 3-4]. While Lyapunov's stability analysis is not performed within the scope of this dissertation, the information provided here is for the reader to understand the complexity of the stability of the ODE system.

Before discussing Lyapunov's method, several terms need to be defined first: autonomous systems and equilibrium points. Consider Eq. (3.10). This system is an autonomous system, because it does not depend explicitly on time. Systems that depend explicitly on time are called non-autonomous systems. Now consider a set of points $\mathbf{a}^*$. If $\boldsymbol{\mathcal{F}}(\mathbf{a}^*) = 0$, then $\mathbf{a}^*$ are called equilibrium points. These definitions are used to define the types of systems for which Lyapunov's method apply to.

For linear, autonomous systems, Lyapunov's first or linearization method is used. This method is a subset of Lyapunov's direct method, which is for nonlinear systems. Consider a linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{4.1}$$

where $\mathbf{A}$ is an $n \times n$ matrix and $\mathbf{x}$ is a vector of length $n$. This system has an equilibrium point at $\mathbf{x}^* = \mathbf{0}$. All stability analysis using Lyapunov's method is for an equilibrium point at the origin. If $\mathbf{x}^* \neq 0$, then the system can be shifted by substituting in $\mathbf{x} = \mathbf{y} + \mathbf{x}^*$ where $\mathbf{y}^* = 0$ is the new equilibrium point. By Lyapunov's first method, if the eigenvalues of $\mathbf{A}$ have all negative real parts for small perturbations around the equilibrium point, then the equilibrium point is stable. Furthermore, if the system remains stable in a ball with radius $R$ centered about the equilibrium point, $B_{R_0}$, then $B_{R_0}$ is a domain of attraction of the equilibrium point. Any initial condition that begins within this domain will not leave it. When $R$ goes to infinity, then the system is globally asymptotically stable.

Lyapunov's first method can be extended to nonlinear systems by linearizing the system about an equilibrium point. Take the nonlinear system in Eq. (3.10). Its linearization is

$$\boldsymbol{\mathcal{F}}(\mathbf{a}) \approx \boldsymbol{\mathcal{F}}(\mathbf{a}^*) + \mathbf{J}(\mathbf{a}^*)(\mathbf{a} - \mathbf{a}^*) \tag{4.2}$$

where $\mathbf{J}(\mathbf{a}^*)$ is the $m \times m$ Jacobian of the ODE system evaluated at the equilibrium point. If one defines $\mathbf{u} = \mathbf{a} - \mathbf{a}^*$, then the linearized ODE becomes

$$\dot{\mathbf{u}} = \mathbf{J}(\mathbf{a}^*)\mathbf{u}. \tag{4.3}$$

Similar to Eq. (4.1), the stability of this system is performed by evaluating the eigenvalues of its matrix $\mathbf{J}(\mathbf{a}^*)$.

Some limitations exist when applying Lyapunov's first method to nonlinear systems. First, the approximation given in Eq. (4.3) may not be accurate to the actual model, in which case the stability

analysis cannot be performed. Second, if the system is non-autonomous and has positive real parts in its eigenvalues, then the stability of the system is inconclusive. Third, the linearization may lose information about the nonlinear system that may be important. To counter these issues, Lyapunov's direct method should be used.

Lyapunov's direct method is based on the dissipation of energy within a system. Consider a pendulum displaced a certain height. At this state it has some stored potential energy. When the pendulum is dropped, the potential energy converts to kinetic energy. After both the kinetic energy and potential energy reduce to zero, the pendulum is at equilibrium. In this case the rate of decay of the energy in the system relates to its stability. In Lyapunov's direct method, an energy function, $V(\mathbf{x})$, is chosen such that $V(\mathbf{x})$ is smooth, $V(\mathbf{x}) > 0$ or positive definite, and $V(\mathbf{0}) = 0$. Similar to the pendulum, stability is determined by analyzing its time derivative at the equilibrium point $\dot{V}(\mathbf{x})$. If $\dot{V}(\mathbf{x}^*) \leq 0$, then the equilibrium point is stable (asymptotically stable if $\dot{V}(\mathbf{x}^*) < 0$. If $\dot{V}(\mathbf{x}^*) > 0$, then the equilibrium point is unstable, because the energy is increasing in the system.

For non-autonomous systems, Lyapunov's direct method may be used given some limitations. The Lyapunov function is now given by a time-varying function $V(\mathbf{x}, t)$. This function is locally positive definite if $\forall\ t \geq t_0, V(\mathbf{x}, t) \geq V_0(\mathbf{x})$ where $V_0(\mathbf{x})$ is a time-invariant positive definite function. Furthermore, this function is decrescent if $V(\mathbf{0}, t) = 0$ and $\forall\ t \geq 0, V(\mathbf{x}, t) \leq V_l(\mathbf{x})$ where $V_l(\mathbf{x})$ is a time-invariant positive definite function. For $\dot{V}(\mathbf{x}, t)$ to be negative semi-definite, then $-\dot{V}(\mathbf{x}, t)$ must be positive semi-definite. If $V(\mathbf{x}, t)$ is positive definite and decrescent and $\dot{V}(\mathbf{x}, t)$ is negative semi-definite in a ball $B_{R_0}$ around the equilibrium point $\mathbf{x}^* = 0$, then $\mathbf{x}^* = 0$ is uniformly stable. Lyapunov's first method can also be applied to non-autonomous systems as well [69, pp. 116-117].

For systems undergoing a disturbance,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{g}(\mathbf{x}, t) \tag{4.4}$$

where $\mathbf{g}(t)$ is the disturbance, then the stability of the ODE system depends on the unperturbed state,

$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ and the size of the perturbation. If $\mathbf{x}^* = 0$ is an equilibrium point of the unperturbed state, and $\|\mathbf{x}(t_0)\| < \delta_1$ and $\|\mathbf{g}(x, t)\| < \delta_2$ about $\mathbf{x}^* = 0$, where $\delta_1$ and $\delta_2$ are scalar numbers, then $\|\mathbf{x}(t)\| < \epsilon$, that is, the equilibrium point is totally stable. Conversely, if the unperturbed state is uniformly asymptotically stable in the sense of Lyapunov, then the perturbed state for small disturbances is totally stable.

### 4.1.2 Penalty Method[1]

The boundary conditions imposed on the FOM constrain the basis functions such that, for the on-reference cases, the ROM will satisfy the boundary conditions. For the off-reference cases, however, the ROM might not satisfy the boundary conditions because the basis functions might not be able to reconstruct the off-reference boundary conditions. If the basis functions do not satisfy the boundary conditions of the FOM, the boundary conditions may end up becoming compatibility constraints on the ROM [30]. These constraints limit the combination of basis functions and time coefficients that can be used for the reconstruction (2.1).

To correct the compatibility constraints, boundary conditions are implemented into the ROM using the penalty method [42, 70]. The penalty method imposes a prescribed boundary condition onto the ODEs of the ROM (3.10). Let $N_{\text{BC}}$ be the number of nodes at the selected boundary. The difference between the ROM solution $\mathbf{Z}_{\text{BC}}^{Z_k} \in \mathbb{R}^{N_{\text{BC}}}, k \in [1, D+2]$, a subset of $\mathbf{Z}$ that includes only the $N_{\text{BC}}$ nodes of the selected boundary of the $k$-th state variable, and the prescribed boundary condition, $\mathbf{F}^{Z_k}(t) \in \mathbb{R}^{N_{\text{BC}}}, k \in [1, D+2]$, is a measure of the boundary condition error. Let $\phi_{\text{BC}i}^{Z_k} \in \mathbb{R}^{N_{\text{BC}}}, i \in [1, n^{Z_k}], k \in [1, D+2]$ be a subset of the POD basis function $\phi_i^{Z_k}, i \in [1, n^{Z_k}], k \in [1, D+2]$, which includes only the $N_{\text{BC}}$ nodes of the boundary of the $k$-th state variable. Let $\mathbf{K}_i^{Z_k}, k \in [1, D+2]$ define the boundary error $\mathbf{Z}_{\text{BC}}^{Z_k} - \mathbf{F}^{Z_k}$ projected on $\phi_{\text{BC}i}^{Z_k}$, $\mathbf{K}_i^{Z_k} = \left(\mathbf{Z}_{\text{BC}}^{Z_k} - \mathbf{F}^{Z_k}, \phi_{\text{BC}_i}^{Z_k}\right), i \in [1, n^{Z_k}], k \in [1, D+2]$. Let $\mathbf{K} \in \mathbb{R}^m$ be the collection of all $\mathbf{K}_i^{Z_k}$ and let $\boldsymbol{\tau} \in \mathbb{R}^{m \times m}$ be a diagonal matrix consisting of the penalty parameters for each ODE of (3.10). The penalty method adds to

[1]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

(3.10) the projection of the boundary error $\mathbf{K}$ scaled by the penalty parameter $\boldsymbol{\tau}$

$$\dot{\mathbf{a}} = \boldsymbol{\mathcal{F}}(\mathbf{a}) - \boldsymbol{\tau}\mathbf{K}. \tag{4.5}$$

A component of the vectorial equation (4.5) is

$$\dot{a}_i^{Z_k} = \mathcal{F}_i^{Z_k}(\mathbf{a}) - \tau_{ii}^{Z_k}\left(\mathbf{Z}_{\text{BC}}^{Z_k} - \mathbf{F}^{Z_k}(t), \phi_i^{Z_k}(\mathbf{x}_{\text{BC}})\right), \quad i \in [1, n^{Z_k}],\ k \in [1, D+2]$$

where the superscript $Z_k$ indicates the $k^{\text{th}}$ component of the state variable and

$$\tau_{ii}^{Z_k} = \tau_{\mathcal{K}\mathcal{K}}, \quad \mathcal{K} = \sum_{j=1}^{k-1} n^{Z_j} + i.$$

The prescribed boundary condition $\mathbf{F}(t) \in \mathbb{R}^{(D+2)N_{\text{BC}}}$ consists of the components

$$\mathbf{F}(t) = \left(\mathbf{F}^{\zeta}, \mathbf{F}^{u}, \mathbf{F}^{v}, \mathbf{F}^{w}, \mathbf{F}^{p}\right)^{T}.$$

The value of the penalty parameter $\tau_{ii}^{Z_k}$ is typically set to some large number. Rather than randomly guessing its value, two methods are explored here for the computation of $\tau_{ii}^{Z_k}$: an energy-based stabilizing method and a method that minimizes the error $\mathbf{K}$ to zero.

The energy-based stabilizing method [42, 43] ensures that the system energy is decreasing. Assuming $\boldsymbol{\tau} = \tau\mathbf{I}$ in (4.5) yields

$$\dot{\mathbf{a}} = \boldsymbol{\mathcal{F}}(\mathbf{a}) - \tau\mathbf{I}\mathbf{K}. \tag{4.6}$$

One can then project (4.6) along $\mathbf{a}$ to get

$$(\dot{\mathbf{a}}, \mathbf{a}) = (\boldsymbol{\mathcal{F}}(\mathbf{a}), \mathbf{a}) - \tau\,(\mathbf{K}, \mathbf{a})\,.$$

Note that $(\dot{\mathbf{a}}, \mathbf{a}) = \frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{a}\|^2$, which represents the time rate of system energy. Bounds for the penalty

parameter are found by ensuring that system energy is decreasing, $\frac{1}{2}\frac{\partial}{\partial t}\left\|\mathbf{a}\right\|^2 \leq 0$, such that

$$\tau \geq \frac{(\boldsymbol{\mathcal{F}}(\mathbf{a}), \mathbf{a})}{(\mathbf{K}, \mathbf{a})}.$$

Alternately, one can calculate the penalty parameter such that the error at the next time step $\mathbf{K}$ is reduced to zero. Let us assume $\boldsymbol{\tau} = \hat{\boldsymbol{\tau}}$, where $\hat{\boldsymbol{\tau}} \in \mathbb{R}^{m \times m}$ is a diagonal matrix with $\hat{\tau}_{i_k} = \hat{\tau}_k,\ i_k \in [1, n^{Z_k}],\ k \in [1, D+2]$, that is, the value of the penalty parameters are held constant for all modes $i_k$ in their respective governing equation, $k$. The value of $\hat{\boldsymbol{\tau}}$ that sets $\mathbf{K}$ to zero is found by using an iterative procedure based on Newton's method such that $\hat{\boldsymbol{\tau}}$ is the root of $\mathbf{K}(\hat{\boldsymbol{\tau}})$

$$\hat{\tau}_{Z_k}^{n+1} = \hat{\tau}_{Z_k}^{n} - \frac{K(\hat{\tau}_{Z_k}^{n})}{dK(\hat{\tau}_{Z_k}^{n})/d\hat{\tau}}, \quad k \in [1, D+2]. \tag{4.7}$$

If $dK(\hat{\tau}_{Z_k}^{n})/d\hat{\tau}$ is approximated using a first-order difference, then (4.7) becomes

$$\hat{\tau}_{Z_k}^{n+1} = \hat{\tau}_{Z_k}^{n} - \frac{K(\hat{\tau}_{Z_k}^{n})}{K(\hat{\tau}_{Z_k}^{n}) - K(\hat{\tau}_{Z_k}^{n-1})}\left(\hat{\tau}_{Z_k}^{n} - \hat{\tau}^{n-1}\right), \quad k \in [1, D+2]. \tag{4.8}$$

Numerical experiments showed that $K(\hat{\tau}_{Z_k}^{n})$ varied linearly with $\hat{\tau}_{Z_k}^{n}$, which led to a fast convergence of the method. While this method does not guarantee stability, it does add to the accuracy of the solution. This method for determining the penalty parameter $\boldsymbol{\tau}$ by finding the roots of $\mathbf{K}$ was used herein instead of the energy-decreasing method because it reliably generated stable results.

#### *4.1.2.1 Note on Lifting Method*

Another possible method that was considered was the "lifting" method. This method incorporates the unsteadiness of the flow solution into a variable that is then substituted into the governing equations. For example, consider Poisson's equation. Let $p(t)$ be the solution to

$$-\Delta p(t) = f \in \Omega,$$
$$p(t) = \bar{p}g(t) \in \partial\Omega$$

where $\bar{p}$ is the solution to the steady-state problem and $g(t)$ is the variation of the pressure at the outlet. Let $\ell$ be the solution to

$$-\Delta\ell = f \in \Omega,$$
$$\ell = \bar{p} \in \partial\Omega$$

where $\ell$ is parameter independent.

Rather than calculate the POD basis functions of $p(t)$, the POD basis functions of a new variable $w(t)$, denoted by $w(t) = p(t) - g(t)\ell$, is found. The governing equations are rewritten with $w(t)$ by substituting in $p(t) = w(t) + g(t)\ell$. For example, the conservation of *x*-momentum equation in one-dimension is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \zeta\frac{\partial p}{\partial x} + \zeta g(t)\frac{\partial \bar{p}}{\partial x} = 0$$

and the conservation of energy equation in one-dimension is

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + \gamma w\frac{\partial u}{\partial x} + \bar{p}\frac{\partial g(t)}{\partial t} + ug(t)\frac{\partial \bar{p}}{\partial x} + \gamma g(t)\frac{\partial u}{\partial x} = 0.$$

The basis functions of $w(t)$ should be zero at the boundaries for which the unsteady boundary conditions are placed. This removes the compatibility constraint on the ROM. Results of this approach are shown in sec. 5.1. It was found that the penalty method was better at stabilizing the ROM than the lifting method, because the lifting method does not address the destabilizing effect of the projection procedure. The penalty method addresses the destabilizing effect by working *a posteriori* of the projection procedure.

### 4.1.3 Artificial Dissipation[2]

Artificial dissipation has been used previously to stabilize POD-based reduced-order models by dissipating spurious modes [36, 37]. The artificial dissipation terms are projected along their

[2]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

respective subspace and added to the system of ODEs of the ROM such that (3.10) becomes

$$\dot{\mathbf{a}} = \boldsymbol{\mathcal{F}}(\mathbf{a}) + \tilde{\boldsymbol{\nu}}\mathbf{S} \tag{4.9}$$

where $\tilde{\boldsymbol{\nu}} \in \mathbb{R}^{m\times m}$ is the diagonal matrix of the artificial dissipation parameters, and $\mathbf{S} \in \mathbb{R}^m$ consists of the projection of the Laplacian of the basis functions along their respective subspace. It should be noted that artificial dissipation is not applied to the zeroth (or mean) mode. For a three-dimensional problem, (4.9) can be written in component form as

$$\dot{a}_i^{Z_k} = \mathcal{F}_i^{Z_k} + \tilde{\nu}_{ii}^{Z_k} \sum_{j=1}^{n^{Z_k}} \left( \phi_{j,xx}^{Z_k} + \phi_{j,yy}^{Z_k} + \phi_{j,zz}^{Z_k}, \phi_i^{Z_k} \right) a_j^{Z_k},$$

$$1 \leq i \leq n^{Z_k},\ 1 \leq k \leq 5$$

where the artificial dissipation parameter, $\tilde{\nu}_{ii}$, must be greater than or equal to zero.

The value of the artificial dissipation parameter $\tilde{\nu}_{ii}$ was found using an energy-based stability analysis. Therefore, artificial dissipation was added to (4.5) to yield

$$\dot{\mathbf{a}} = \boldsymbol{\mathcal{F}}(\mathbf{a}) + \tilde{\boldsymbol{\nu}}\mathbf{S} - \boldsymbol{\tau}\mathbf{K}. \tag{4.10}$$

Assuming $\tilde{\boldsymbol{\nu}} = \tilde{\nu}\mathbf{I}$ and projecting along $\mathbf{a}$ gives

$$(\dot{\mathbf{a}}, \mathbf{a}) = (\boldsymbol{\mathcal{F}}(\mathbf{a}), \mathbf{a}) + \tilde{\nu}\,(\mathbf{S}, \mathbf{a}) - (\boldsymbol{\tau}\mathbf{K}, \mathbf{a})$$

where $(\dot{\mathbf{a}}, \mathbf{a}) = \frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{a}\|^2$ is the system energy, which should be decreasing for stability, *i.e.*, $\frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{a}\|^2 \leq 0$. Consequently, the dissipation must satisfy the inequalities

$$0 \leq \tilde{\nu} \leq \frac{(\boldsymbol{\tau}\mathbf{K} - \boldsymbol{\mathcal{F}}(\mathbf{a}), \mathbf{a})}{(\mathbf{S}, \mathbf{a})}. \tag{4.11}$$

### 4.1.4 Modified Number of Modes[3]

The size of the system of ODEs (3.10) is $m = \sum_{k=1}^{D+2} n^{Z_k}$. Consequently, the value of $m$ depends on the number of modes $n^\zeta$ used to approximate the $\zeta$ variable in the mass balance equation, the number of modes $n^u$ used to approximate velocity $u$ in the $x$-component of the momentum conservation equation, and so on for the $v$, $w$ and $p$ variables. If the number of modes used to approximate variable $u$ in the mass balance equation is less than $n^u$, the size of the system of ODEs is not changed. The Jacobian of the reduced-order model, however, is changed. Consequently, the eigenvalues of the Jacobian are changed, and this change affects the stability of the reduced-order model.

For example, consider the projection of the conservation of mass equation for one-dimensional flow

$$\dot{a}_k^\zeta + \sum_{i=0}^{n^\zeta} \sum_{j=0}^{n^u} \left( \phi_{i,x}^\zeta \phi_j^u - \phi_i^\zeta \phi_{j,x}^u, \phi_k^\zeta \right) a_i^\zeta a_j^u = 0, \; k \in [1, n^\zeta] \tag{4.12}$$

Assuming $n^\zeta = 1$ and $n^u = 2$ in (4.12) yields

$$\begin{aligned} \dot{a}_k^\zeta &+ \left( \phi_{0,x}^\zeta \phi_0^u - \phi_0^\zeta \phi_{0,x}^u, \phi_k^\zeta \right) + \left( \phi_{0,x}^\zeta \phi_1^u - \phi_0^\zeta \phi_{1,x}^u, \phi_k^\zeta \right) a_1^u + \\ &+ \left( \phi_{0,x}^\zeta \phi_2^u - \phi_0^\zeta \phi_{2,x}^u, \phi_k^\zeta \right) a_2^u + \left( \phi_{1,x}^\zeta \phi_0^u - \phi_1^\zeta \phi_{0,x}^u, \phi_k^\zeta \right) a_1^\zeta + \\ &+ \left( \phi_{1,x}^\zeta \phi_1^u - \phi_1^\zeta \phi_{1,x}^u, \phi_k^\zeta \right) a_1^\zeta a_1^u + \left( \phi_{1,x}^\zeta \phi_2^u - \phi_1^\zeta \phi_{2,x}^u, \phi_k^\zeta \right) a_1^\zeta a_2^u = 0. \end{aligned} \tag{4.13}$$

If $n^u$ is set equal to one within this equation only, then (4.13) reduces to

$$\begin{aligned} \dot{a}_k^\zeta &+ \left( \phi_{0,x}^\zeta \phi_0^u - \phi_0^\zeta \phi_{0,x}^u, \phi_k^\zeta \right) + \left( \phi_{0,x}^\zeta \phi_1^u - \phi_0^\zeta \phi_{1,x}^u, \phi_k^\zeta \right) a_1^u + \\ &+ \left( \phi_{1,x}^\zeta \phi_0^u - \phi_1^\zeta \phi_{0,x}^u, \phi_k^\zeta \right) a_1^\zeta + \left( \phi_{1,x}^\zeta \phi_1^u - \phi_1^\zeta \phi_{1,x}^u, \phi_k^\zeta \right) a_1^\zeta a_1^u = 0. \end{aligned}$$

Note that $n^u$ cannot be changed within the conservation of *x*-momentum equation, and $n^u$ can only be reduced within the other equations.

[3]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

The effect of modifying the number of modes is apparent when analyzing the Jacobian of the reduced-order model

$$J_{ij}^{Z_k} = \frac{\partial \mathcal{F}_i^{Z_k}(\mathbf{a})}{\partial a_j}, \quad i,j \in [1, n^{Z_k}],\ k \in [1, D+2].$$

The part of the Jacobian that accounts for the effect of $x$-velocity $u$ on the mass balance equation is

$$\frac{\partial \boldsymbol{\mathcal{F}}^{\zeta}}{\partial \mathbf{a}^u} = \begin{bmatrix} \frac{\partial \mathcal{F}_1^{\zeta}}{\partial a_1^u} & \frac{\partial \mathcal{F}_1^{\zeta}}{\partial a_2^u} & \cdots & \frac{\partial \mathcal{F}_1^{\zeta}}{\partial a_{n^u}^u} \\ \frac{\partial \mathcal{F}_2^{\zeta}}{\partial a_1^u} & \frac{\partial \mathcal{F}_2^{\zeta}}{\partial a_2^u} & \cdots & \frac{\partial \mathcal{F}_2^{\zeta}}{\partial a_{n^u}^u} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{F}_{n^{\zeta}}^{\zeta}}{\partial a_1^u} & \frac{\partial \mathcal{F}_{n^{\zeta}}^{\zeta}}{\partial a_2^u} & \cdots & \frac{\partial \mathcal{F}_{n^{\zeta}}^{\zeta}}{\partial a_{n^u}^u} \end{bmatrix} \tag{4.14}$$

If the velocity $u$ in the mass balance equation is approximated by $n^u - 1$ terms instead of $n^u$, then all the terms of the last column of (4.14) are zero. Consequently, the eigenvalues of the Jacobian are modified. The number of modes is modified in order to remove the spurious modes that have a destabilizing effect on the solution of the system of ODEs.

The decision to modify the number of modes within the governing equations comes by observation only. It can be speculated that doing acts as a filter to remove spurious modes from within the governing equations themselves. These spurious modes only have a destabilizing effect rather than a stabilizing one. It may also help with errors that, although may be small in the FOM, are magnified in the ROM through the projection procedure.

## 4.2 Basis Function Interpolation

The main benefit of a ROM consists in its ability to efficiently predict flow conditions beyond what is predicted by the FOM, that is, at off-reference conditions. To predict off-reference conditions, the basis functions, which are functions of space, need to be interpolated to the new flow conditions. This can be done in two ways: by enriching the snapshot database with solutions of different flow conditions, referred to as global POD, or by interpolating using the tangent space to a Grassmann manifold [45]. Both methods are used in this dissertation.

### 4.2.1 Enriching the Snapshot Database

To enrich the snapshot database, FOM solutions are generated that bound the parameter(s) that need to be explored. For example, say the parameter of interest is the angular velocity $\tilde{\omega}$ of some oscillating boundary condition, and a solution is sought for $\tilde{\omega}_t$. Two FOM solutions are generated with angular velocities $\tilde{\omega}_1$ and $\tilde{\omega}_2$, where $\tilde{\omega}_1 < \tilde{\omega}_t < \tilde{\omega}_2$. An interpolated basis is found by combining the FOM snapshots of the solutions of $\tilde{\omega}_1$ and $\tilde{\omega}_2$, and finding the POD basis of the combined solutions that minimizes (2.3). This interpolated basis can be used by the ROM to solve for any solution with an angular velocity that lies between $\tilde{\omega}_1$ and $\tilde{\omega}_2$. Solutions for angular velocities outside of the interval can also be generated, but the approximation errors typically increase [11].

Ideally, the accuracy of the interpolated basis will increase when increasing the number of snapshots appended to the snapshot database. One can also append snapshots with multiple sets of parameters. In this sense, a large enriched snapshot database has the capability of providing a plethora of solutions without having to rerun the FOM.

### 4.2.2 Grassmann Interpolation

Basis interpolation using the tangent space to a Grassmann manifold was developed by Amsallem [45] for POD-based reduced-order models. A further example of its use for off-reference conditions is given in [46].

For a set of flow variables $\mathbf{q} \in \mathbb{R}^{D+2}$, one can obtain a set of POD basis modes $\boldsymbol{\phi} \in \mathbb{R}^{(D+2)\times N}$. The POD basis $\boldsymbol{\phi}$ lies on a subspace $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{D+2}$ with dimension $N$. These subspaces belong to, or are points of, a Grassmann manifold $\boldsymbol{\mathcal{G}}(D+2, N)$. To interpolate to a separate subspace, one can use the tangent space to $\boldsymbol{\mathcal{S}}$ with points $\boldsymbol{\Gamma} \in \mathbb{R}^{(D+2)\times N}$. The subspaces themselves form a subset of orthonormal matrices belonging to a compact Stiefel manifold, which is a unit sphere manifold. Therefore, to interpolate to a new subspace, one can use the geodesic path on the manifold, which requires only a reference point and an initial derivative, *i.e.*, an initial direction. Because the interpolation is done in the manifold, the constraints on the original space are preserved, *i.e*, the

orthogonality of the bases.

Given $N_R$ sets of reference subspaces with respective operating points or flow parameters $\{\tilde{\lambda}_i\}_{i=0}^{N_R-1}$ associated with them, where $\tilde{\lambda}_{N_R}$ is the desired operating point of the new subspace, one can find the interpolated set of basis functions $\boldsymbol{\phi}_{N_R}$. A thin Singular Value Decomposition (SVD) is taken of a product of the bases

$$\left(\mathbf{I} - \boldsymbol{\phi}_{i_0}\boldsymbol{\phi}_{i_0}^T\right)\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_{i_0}^T\boldsymbol{\phi}_i\right)^{-1} = \mathbf{U}_i\boldsymbol{\Sigma}_i\mathbf{V}_i^T,\ i = 0, N_R - 1,\ i \neq i_0$$

where $\boldsymbol{\phi}_{i_0}$ is a reference basis, which is the basis with operating point closest to the desired operating point $\tilde{\lambda}_{N_R}$. The tangent subspaces of the known sets of bases are found by using the results of the thin SVD.

$$\boldsymbol{\Gamma}_i = \mathbf{U}_i \tan^{-1}\left(\boldsymbol{\Sigma}_i\right)\mathbf{V}_i^T,\ i = 0, N_R - 1,\ i \neq i_0$$

A Lagrangian interpolation with coefficients $\tilde{r}_i$ given by

$$\tilde{r}_i = \prod_{j \neq i} \frac{\tilde{\lambda}_0 - \tilde{\lambda}_j}{\tilde{\lambda}_i - \tilde{\lambda}_j},\ i = 1, N_R - 1,\ i \neq i_0 \tag{4.15}$$

is used to interpolate along the tangent subspace $\boldsymbol{\Gamma}$ to find the tangent subspace at the desired operating point

$$\boldsymbol{\Gamma}(\tilde{\lambda}_{N_R}) = \sum_{i=0}^{N_R-1} \tilde{r}_i\boldsymbol{\Gamma}_i(\tilde{\lambda}_i),\ i = 0, N_R - 1,\ i \neq i_0.$$

To obtain the interpolated basis, a thin SVD of the tangent subspace at the desired operating point is performed $\boldsymbol{\Gamma}(\tilde{\lambda}_{N_R}) = \mathbf{U}_{N_R}\boldsymbol{\Sigma}_{N_R}\mathbf{V}_{N_R}^T$ where the interpolated basis is

$$\boldsymbol{\phi}_{N_R} = \boldsymbol{\phi}_{i_0}\mathbf{V}_{N_R}\cos\left(\boldsymbol{\Sigma}_{N_R}\right) + \mathbf{U}_{N_R}\sin\left(\boldsymbol{\Sigma}_{N_R}\right). \tag{4.16}$$

Note that the basis corresponding to the zeroth mode or mean $\boldsymbol{\phi}_{N_R}^0 \in \mathbb{R}^{(D+2)}$ is found using a Lagrangian interpolation $\boldsymbol{\phi}_{N_R}^0 = \sum_{i=0}^{N_R} r_i\boldsymbol{\phi}_i^0,\ i \neq i_0$.

If there are multiple variables to consider when interpolating, one can instead replace the

Lagrangian interpolation with any multivariate interpolation method. For example, one can use bivariate interpolation [71]. The author notes that using a multivariate interpolation scheme will increase the offline costs, because it requires more FOM snapshots to interpolate off of.

### 4.3 ODEPACK[4]

The governing system of ODEs of the ROM (3.10) was solved using ODEPACK [72]. The relative and absolute tolerance parameters in ODEPACK were set to 0.1 and 0.001, respectively. The DLSODE solver was selected. This solver has both nonstiff and stiff solvers imbedded into it. The nonstiff solver uses Adams' method while the stiff solver uses backward differentiation formulas. The Jacobian was generated internally. The initial conditions were taken from time coefficients derived from a FOM solution (2.9). A least-squares with QR decomposition method was used to evaluate the spatial derivatives in the ROM [73, p. 165-168].

[4]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

# 5. VERIFICATION AND VALIDATION OF ZETA-ROM

This section presents verification and validation results of the zeta-ROM for both on- and off-reference conditions. The cases selected were to test different aspects of the zeta-ROM throughout its development, *e.g.*, its stability and its ability to accurately reconstruct and predict flow solutions of varying unsteadiness.

## 5.1 Quasi-One-Dimensional Nozzle

This section presents results for inviscid flow through a quasi-one-dimensional nozzle. The quasi-one-dimensional nozzle is a diverging nozzle that has an area variation $A(x)$ [58, p. 327]

$$A(x) = \begin{cases} 1 + 2.2(x - 0.5)^2 & 0.0 \leq x \leq 0.5 \\ 1 + 0.2223(x - 0.5)^2 & 0.5 \leq x \leq 1.0 \end{cases}. \tag{5.1}$$

The 1D FOM was used to generate the snapshots for the ROM. It is important to note that while the governing equations of the FOM used characteristic variables, the ROM used the specific volume formulation of the inviscid form of the governing equations (3.6). A grid convergence study was performed for the steady flow through the nozzle by using 51, 101, and 201 nodes. Table 5.1 shows the results of the study where the percentages are the percent change of the steady state Mach number between grid refinements. The steady state Mach number approached an asymptotic value as the grid was refined. Due to the small change between the three grids, results are shown for the grid using 51 nodes.

To create an unsteady flow, an oscillating outlet pressure was imposed

$$p(t, x_{\text{out}}) = f(t) = \bar{p}\left(1 + \tilde{A}\sin\left(\tilde{\omega}t + \varphi\right)\right) \tag{5.2}$$

where the subscript “out" denotes values at the outlet boundary, $\bar{p}$ is the mean pressure at the outlet, $\tilde{A}$ is the amplitude of the oscillation, $\tilde{\omega}$ is the angular velocity of the oscillation, and $\varphi$ is the phase

| Grid Points | Steady-State Outlet Mach Number | Change from Previous (%) |
|---|---|---|
| 51 | 0.3598 | - |
| 101 | 0.3603 | 0.16 |
| 201 | 0.3605 | 0.04 |

Table 5.1: Grid convergence of the quasi-one-dimensional nozzle.

shift. Five cases for the ROM were created by varying the amplitude $\tilde{A}$ and angular velocity $\tilde{\omega}$ of the outlet pressure oscillation. Table 5.2 summarizes these cases. Cases 1 to 3 are on-reference

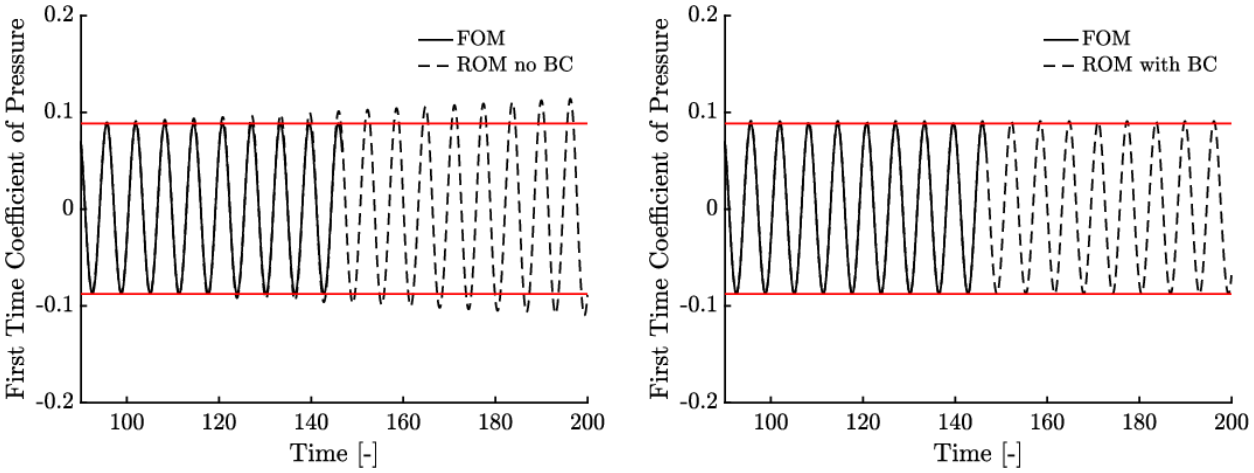
| Case | $\tilde{\omega}$ [rad/sec] | $\tilde{A}$ | Type |
|---|---|---|---|
| 1 | 1.0 | 0.020 | on-reference |
| 2 | 1.0 | 0.030 | on-reference |
| 3 | 2.0 | 0.020 | on-reference |
| 4 | 1.0 | 0.025 | off-reference |
| 5 | 1.5 | 0.020 | off-reference |

Table 5.2: On- and off-reference cases for the quasi-one-dimensional nozzle with oscillating outlet pressure.

cases and cases 4 and 5 are off-reference cases. For all cases $\bar{p} = 0.95$ and $\varphi = 0.141\pi$. The FOM simulated the flow for 8 periods, and a total of 2000 snapshots were saved.

A penalty-enforced boundary condition was imposed onto the ROM. The prescribed boundary condition was applied at the outlet $F(t) = (0, 0, f(t))^T$ where $f(t)$ is the oscillating outlet pressure boundary condition imposed on the FOM in Eq. (5.4). The penalty parameter, $\boldsymbol{\tau}$, was calculated by finding the roots of the error $\mathbf{K}$ (4.8).

For brevity, only case 1 results are shown here for the on-reference cases. Cases 2 and 3 showed similar results. All cases used the following number of modes $n^\zeta = n^u = n^p = 2$. Figure 5.1 shows the stabilizing effect of the penalty-enforced boundary condition on the solution of the first time coefficient of pressure. Both $\zeta$ and $u$ displayed similar results.




(a) ROM without Penalty-Enforced BC

(b) ROM with Penalty-Enforced BC

Figure 5.1: Effect of penalty-enforced boundary condition on ROM solution for a quasi-one-dimensional nozzle case with outlet pressure oscillation with angular velocity $\tilde{\omega} = 1$ rad/sec and amplitude $\tilde{A} = 0.02$. The red lines correspond to the bounds of the time coefficients derived from the FOM solution.

The time coefficients were calculated in three ways: (1) the line corresponding to “FOM" solved Eq. (2.9) for the “exact" time coefficients, (2) the line corresponding to “ROM with no BC" solved Eq. (3.10) for the time coefficients, and (3) the line corresponding to “ROM with BC" solved Eq. (4.5) for the time coefficients. Both ROM solutions were extrapolated past the sampling time of the FOM. The ROM solution without the penalty-enforced boundary condition had an unstable growth over time. The ROM solution with the penalty-enforced boundary condition remained within the bounds of the time coefficients derived from the FOM solution, resulting in a stable solution. The penalty method, therefore, worked in stabilizing the ROM by imposing the prescribed boundary condition. In addition, there was also good agreement between the ROM solution and the FOM solution once the penalty method was imposed.

The phase portrait in Figure 5.2 shows the effect of imposing the boundary conditions onto the ROM for $\zeta$. The ROM solution without the penalty-enforced boundary condition displays an unstable outward spiral while the ROM with the penalty-enforced boundary condition displays a stable Limit Cycle Oscillation (LCO) indicative of its oscillating outlet pressure boundary condition.

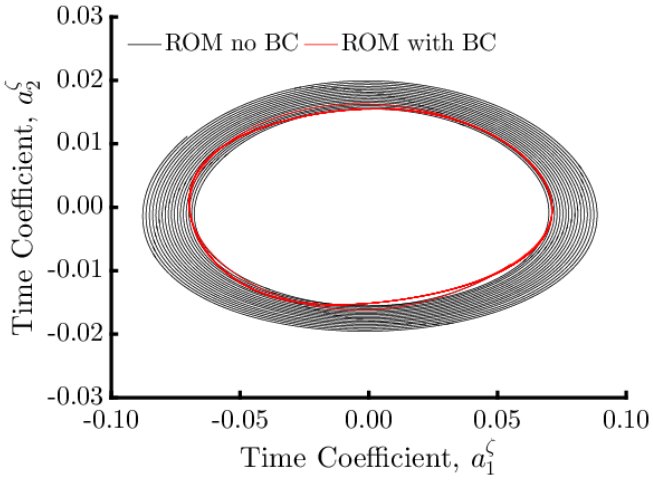
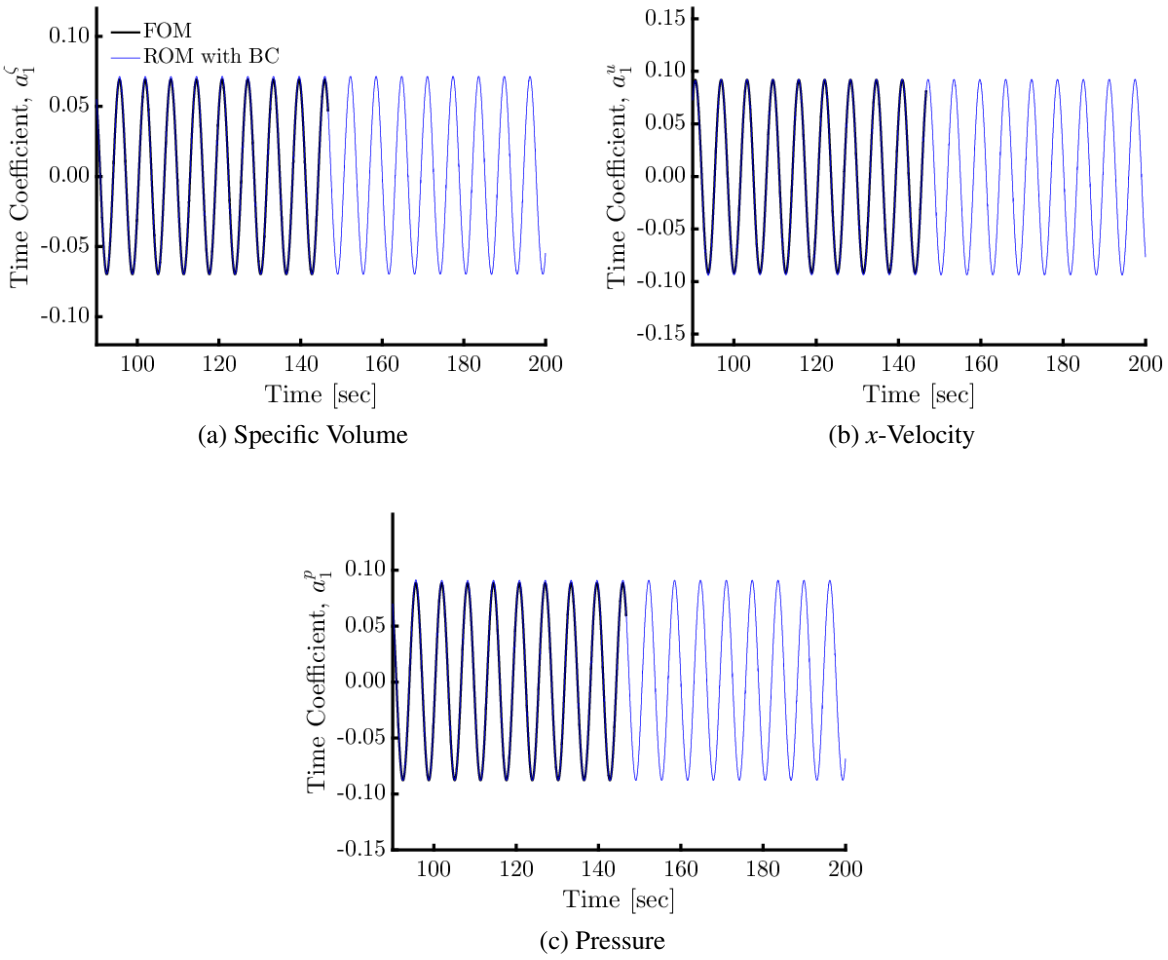



Figure 5.2: Phase portrait of the time coefficients of specific volume for a quasi-one-dimensional nozzle case with outlet pressure oscillation with angular velocity $\tilde{\omega} = 1$ rad/sec and amplitude $\tilde{A} = 0.02$.

To interpolate the basis functions, the snapshots of cases 1 and 2 were combined to develop the autocorrelation matrix (2.5) for case 4. The snapshots of cases 1 and 3 were combined to develop the autocorrelation matrix for case 5. It is important to note that the zeta-variable ROM used the interpolated basis functions developed by enriching the snapshot database to solve for the off-reference solutions. An interesting note is that for case 5, which interpolated frequency, enriching the snapshot database with an unequal number of periods between solutions led to erroneous results from the 1D ROM.

Figures 5.3 shows a comparison of the "exact" time coefficients found by solving (2.9) and the "approximate" time coefficients found by solving the system of ODEs (4.5). For conciseness, only case 1 results are shown for the on-reference cases. All other on-reference cases showed similar results. The zeta-ROM solution is extended past the runtime of the FOM to show the continuous stabilizing effect of the penalty method on the zeta-ROM. Here, the zeta-ROM was able to accurately reconstruct the FOM solution.

The penalty method for the off-reference cases was used in a manner similar to that for the on-reference cases. Consequently, the penalty method was used to constrain the solution of the




(a) Specific Volume

(b) $x$-Velocity

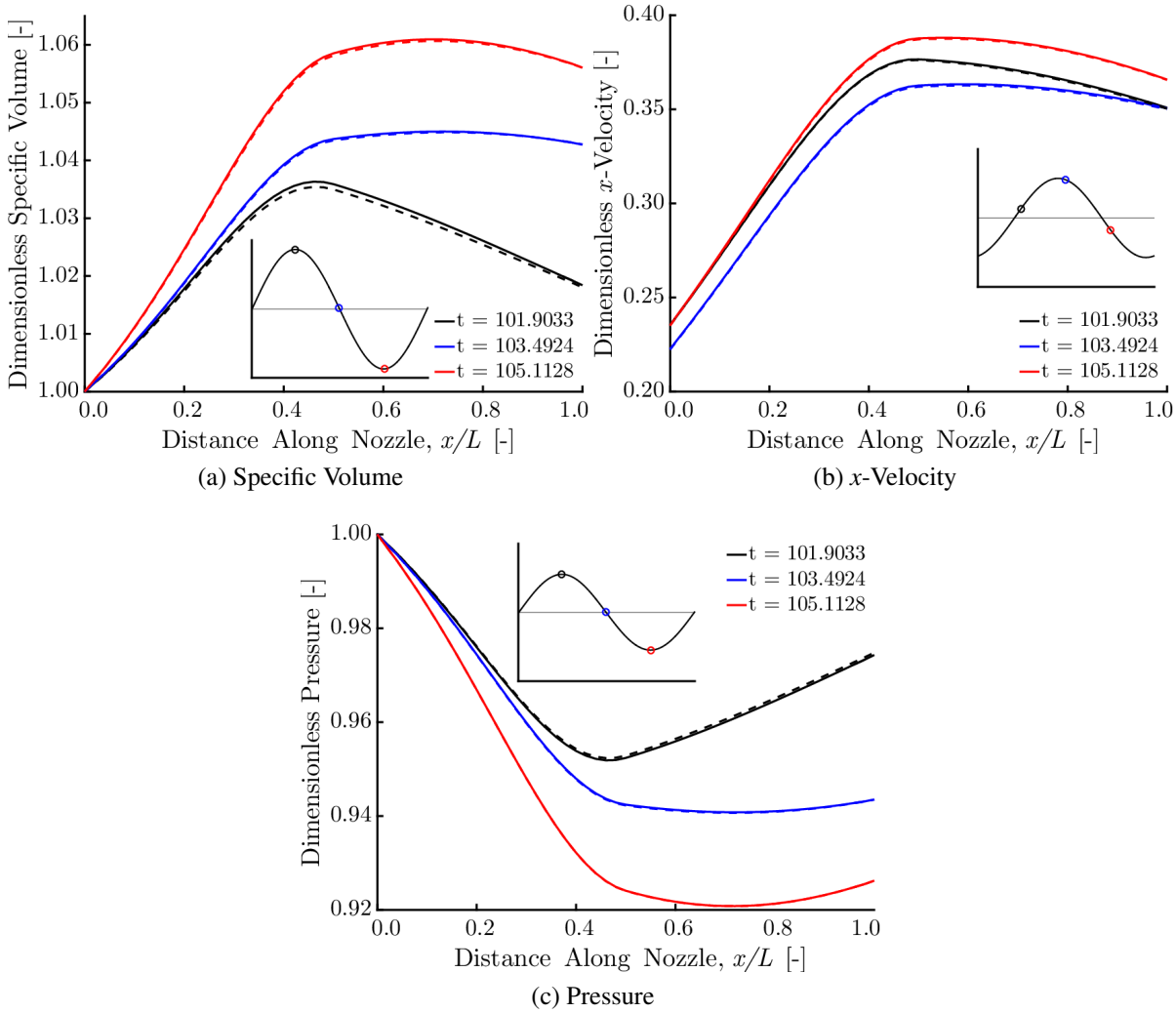
(c) Pressure

Figure 5.3: Comparison of the approximate time coefficients of the zeta-ROM versus the exact time coefficients for a quasi-one-dimensional nozzle with an outlet pressure oscillation with angular velocity $\tilde{\omega}$ = 1 rad/sec and amplitude $\tilde{A} = 0.02$.

zeta-variable ROM such that it matched the desired flow conditions, *i.e.*, the angular velocity and amplitude of the outlet pressure oscillation. Like in case 1, the penalty parameter $\boldsymbol{\tau}$ was determined by finding the roots of the error $\mathbf{K}$ (4.7).

Figures 5.4 and 5.5 show a comparison between the reconstructed zeta-variable ROM solution and the FOM solution for cases 4 and 5, respectively, at three different times corresponding to the minimum, mean, and maximum of the outlet pressure oscillation. It is important to note that the respective FOM solutions were only developed for comparison, and the snapshots of the respective

FOM solutions were not used to develop the autocorrelation matrix. The ROM solution shows good agreement with the respective FOM solution for all three times for both cases 4 and 5.




(a) Specific Volume

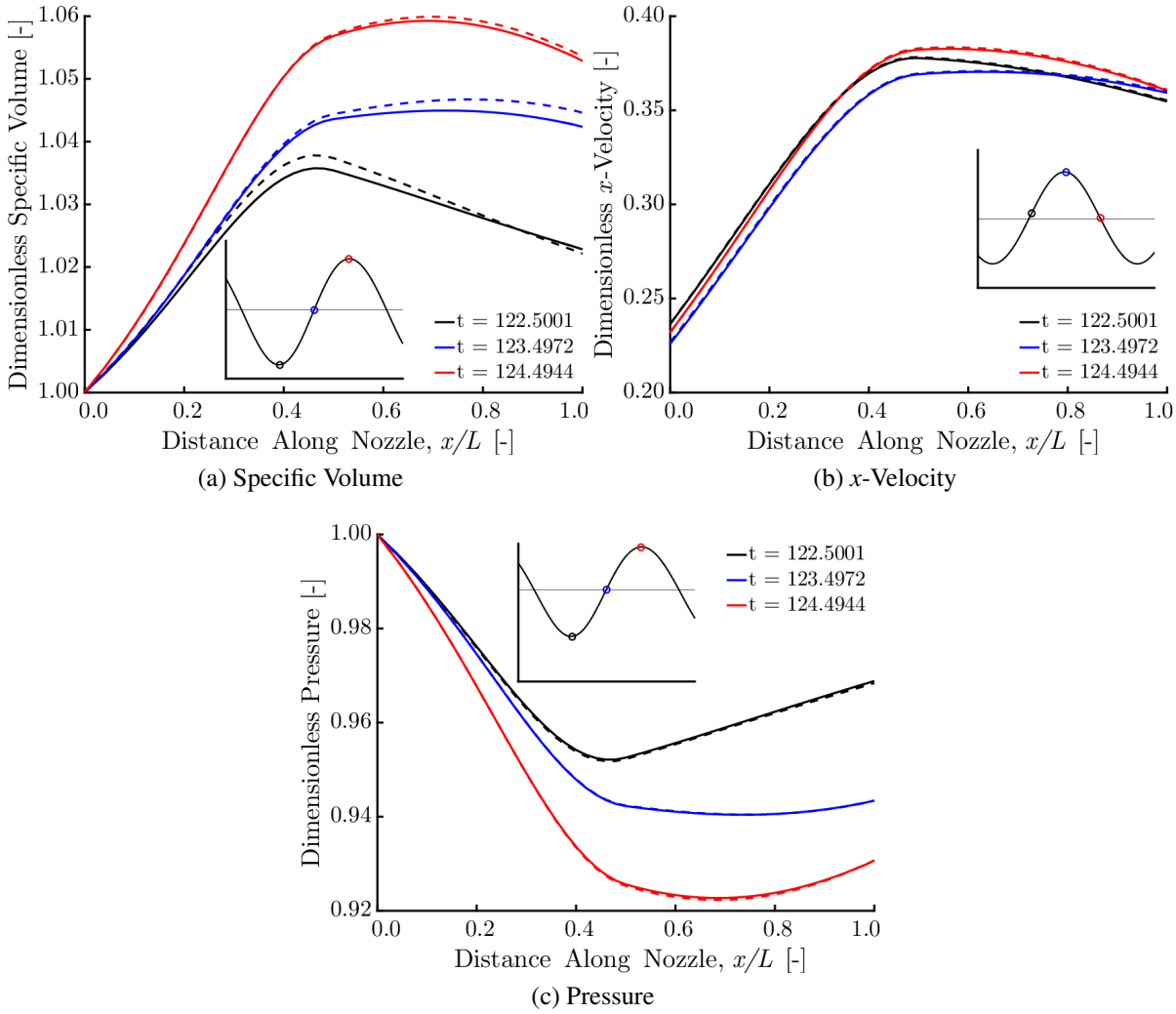
(b) $x$-Velocity

(c) Pressure

Figure 5.4: Off-reference ROM solution vs. FOM solution for a quasi-one-dimensional nozzle with an outlet pressure oscillation with angular velocity $\tilde{\omega}$ = 1 rad/sec and amplitude $\tilde{A} = 0.025$. The dashed lines are the ROM solution and the solid lines are the FOM solution.

### 5.1.1 Effect of Lifting Method

The lifting method was also applied to case 1. The zeta-ROM solved the ODE system with $w(t)$ substituted in. The ROM used the same number of modes as the previous cases. The penalty




(a) Specific Volume

(b) $x$-Velocity

(c) Pressure

Figure 5.5: Off-reference ROM solution vs. FOM solution for a quasi-one-dimensional nozzle with an outlet pressure oscillation with angular velocity $\tilde{\omega}$ = 1.5 rad/sec and amplitude $\tilde{A} = 0.02$. The dashed lines are the ROM solution and the solid lines are the FOM solution.

method was not enforced.

Figure 5.6 shows the results of solving the ODE solution using the lifting approach with the same number of modes as before. The lifting approach enforces the boundary conditions onto the POD-based ROM *a priori* of the projection procedure. Unfortunately, the instabilities of the projection procedure weaken this constraint. This leads to an explosion of the ODE solution when solving. The penalty method was preferred because it enforced the boundary conditions *a posteriori* the projection process. In this way, the instabilities of the projection procedure were addressed, and

a stable solution that adequately satisfied the boundary conditions was generated.

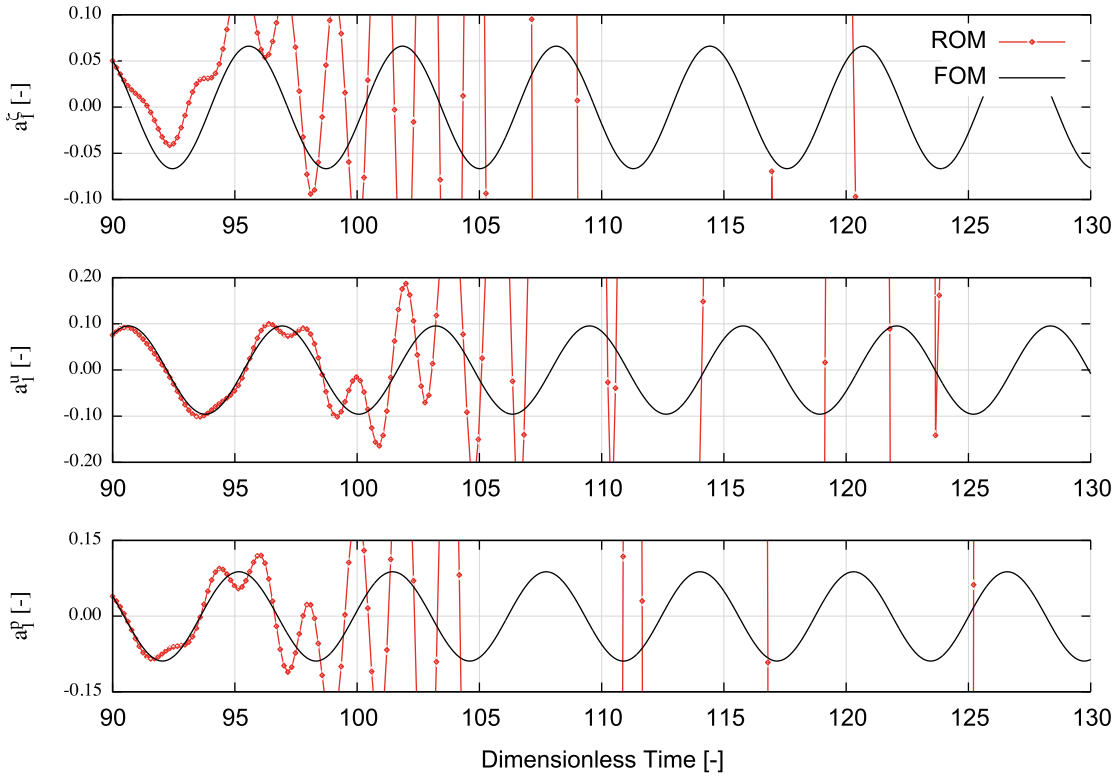



Figure 5.6: Results using the lifting approach on a quasi-one-dimensional nozzle with $\tilde{\omega} = 1$ rad/sec and $\tilde{A} = 0.02$.

## 5.2 Channel Flow[1]

This sections presents results of inviscid flow through a two-dimensional channel. The area variation for the channel is given by [58, p.327]

$$A(x) = \begin{cases} 1 + 2.2(x - 0.5)^2 & 0.0 \leq x \leq 0.5 \\ 1 + 0.2223(x - 0.5)^2 & 0.5 \leq x \leq 1.0 \end{cases} \tag{5.3}$$

[1]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

where the channel has a unit span. The inlet stagnation pressure and temperature are 101,325 Pa and 288.15 K, respectively. The flow enters the inlet along the $x$-axis. The mean pressure at the outlet is 98,538 Pa. The inlet was extended by a quarter of the length of the domain such that the flow enters the nozzle at a zero degree angle. The total length of the channel was 4 m. A cubic spline was used to smooth out the top boundary. Figure 5.7 shows the computational mesh of the channel.

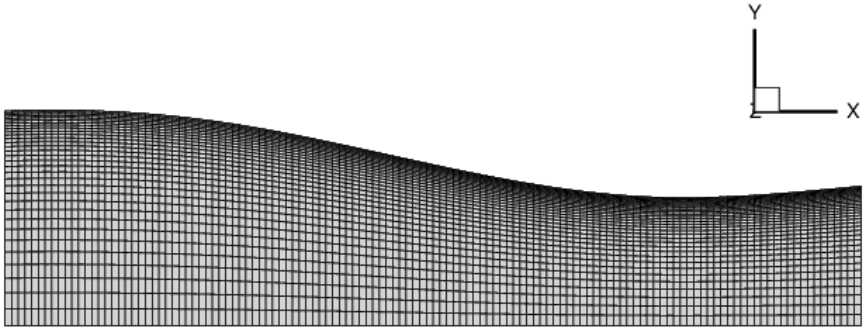



Figure 5.7: Computational mesh of the two-dimensional channel with 4257 nodes.

A grid convergence study was done assuming steady flow. Table 5.3 shows the variation of the average Mach number at the outlet for three grids: coarse, medium and fine with 4257, 16,705, and 66,177 grid nodes, respectively. Table 5.3 also shows the percent change between grid refinements.

| Grid | Steady-State Outlet Mach Number | Change from Previous (%) |
|---|---|---|
| Coarse | 0.1993 | - |
| Medium | 0.1996 | 0.14 |
| Fine | 0.1998 | 0.07 |

Table 5.3: Grid convergence of the two-dimensional channel.

As the grid was refined, the variation of the average Mach number decreased. Since the solution variation with grid size was small, the coarse grid results are discussed in the following paragraphs.

To create an unsteady flow, an oscillating outlet pressure was imposed

$$p(t, x_{\text{out}}) = f(t) = \bar{p}\left(1 + \tilde{A}\sin\left(\tilde{\omega}t\right)\right) \tag{5.4}$$

where the subscript "out" denotes values at the outlet boundary, $\bar{p}$ is the mean pressure at the outlet, $\tilde{A}$ is the amplitude of the oscillation, and $\tilde{\omega}$ is the angular velocity of the oscillation. Three cases were created by varying the amplitude and angular velocity of the outlet pressure oscillation. Table 5.4 summarizes these cases.

| Case | $\tilde{\omega}$ [rad/sec] | $\tilde{A}$ |
|---|---|---|
| 1 | 10 | 0.01 |
| 2 | 10 | 0.02 |
| 3 | 20 | 0.01 |

Table 5.4: On-reference cases for the two-dimensional channel.

For all cases, the pressure was $\bar{p}$ = 98,538 Pa. The full-order model generated snapshots for approximately 3 periods, each period consisting of 300 time steps. A total of 900 snapshots were used to generate the autocorrelation matrix $\mathbf{R}$.

The penalty method was imposed onto the ROM with a prescribed boundary condition applied at the outlet

$$\mathbf{F}(t) = \left(\left(\gamma\mathbf{f}(t)\right)^{\frac{1}{\gamma}}, 0, 0, \mathbf{f}(t)\right)^T$$

where $\mathbf{f}(t) \in \mathbb{R}^{N_{\text{out}}}$ and $N_{\text{out}}$ is the number of nodes at the outlet boundary. In addition, $\mathbf{f}_i(t) = f(t)$ of (5.4), $1 \leq i \leq N_{\text{out}}$. The prescribed boundary condition for $\zeta$ was found through an isentropic relationship [39]. Its derivation is shown in Appendix A. The penalty parameter $\boldsymbol{\tau}$ was solved for using (4.8).

The ROM solution used the following number of modes, $n^{\zeta} = 1$,
$n^u = n^v = n^p = 2$, with a modified number of modes of $n^v = 1$ in the energy conservation

equation. Modifying $n^v$ in the energy conservation equation altered the eigenvalues of the Jacobian of the ROM system such that the ROM was stabilized. Figure 5.8 shows the effect of the modified number of modes on the ROM solution for case 1.

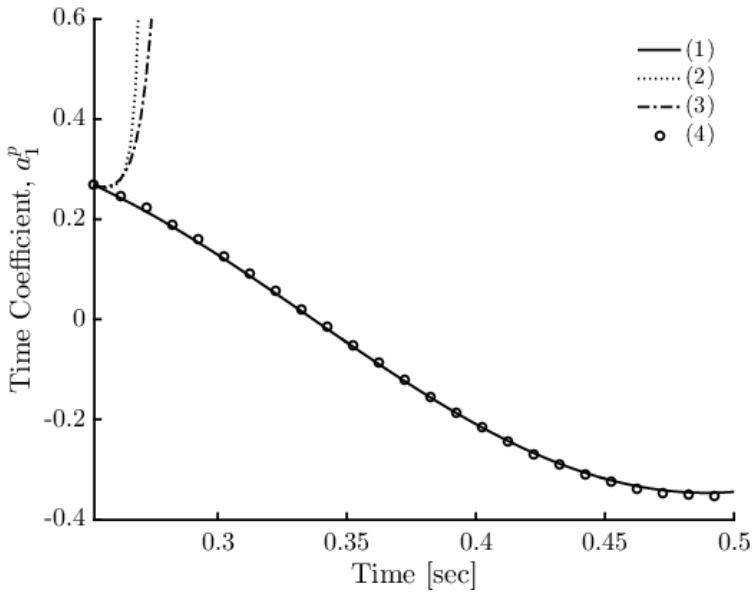



Figure 5.8: Stabilizing effect of the penalty-enforced boundary conditions and modified number of modes on the ROM solution for a two-dimensional channel case with oscillating outlet pressure of angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$: (1) time coefficients derived from the FOM solution, (2) ROM solution, (3) ROM solution with penalty-enforced boundary conditions, (4) ROM solution with penalty-enforced boundary conditions and a modified number of modes.

Once both the penalty method was imposed and the number of modes was modified, the ROM solution agreed well with the "exact" time coefficients (2.9) derived from the FOM without solving the system of ODEs.

For brevity, results are only shown for case 1. The other cases had similar results. The relative error of the ROM solution to the FOM solution

$$\epsilon_i = \frac{|Z_{i,\text{FOM}} - Z_{i,\text{ROM}}|}{|Z_{i,\text{FOM}}|} \tag{5.5}$$

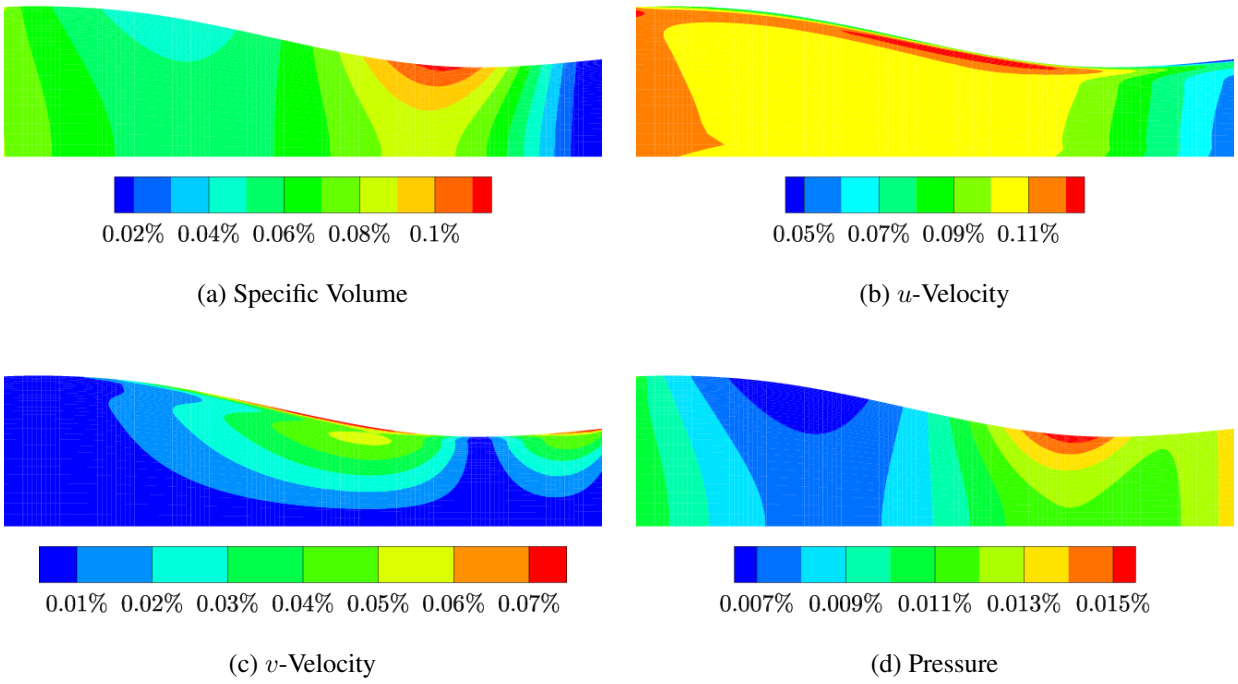
was calculated at each node point to assess the accuracy of the ROM. $Z_i$ is either the first or the

$D$+2 component of the state vector. To avoid division by zero when the velocity components vanish, the relative error for all velocity components was calculated as

$$\epsilon_{j,v} = \frac{|Z_{j,\text{FOM}} - Z_{j,\text{ROM}}|}{|Z_{j,\text{FOM}}| + 1} \tag{5.6}$$

The $j$ index has values between 2 and $D$+1.

Figure 5.9 shows the time-averaged relative error of the ROM solution with respect to the FOM solution for case 1. The ROM solution agreed well with the FOM solution as the error was less than




(a) Specific Volume

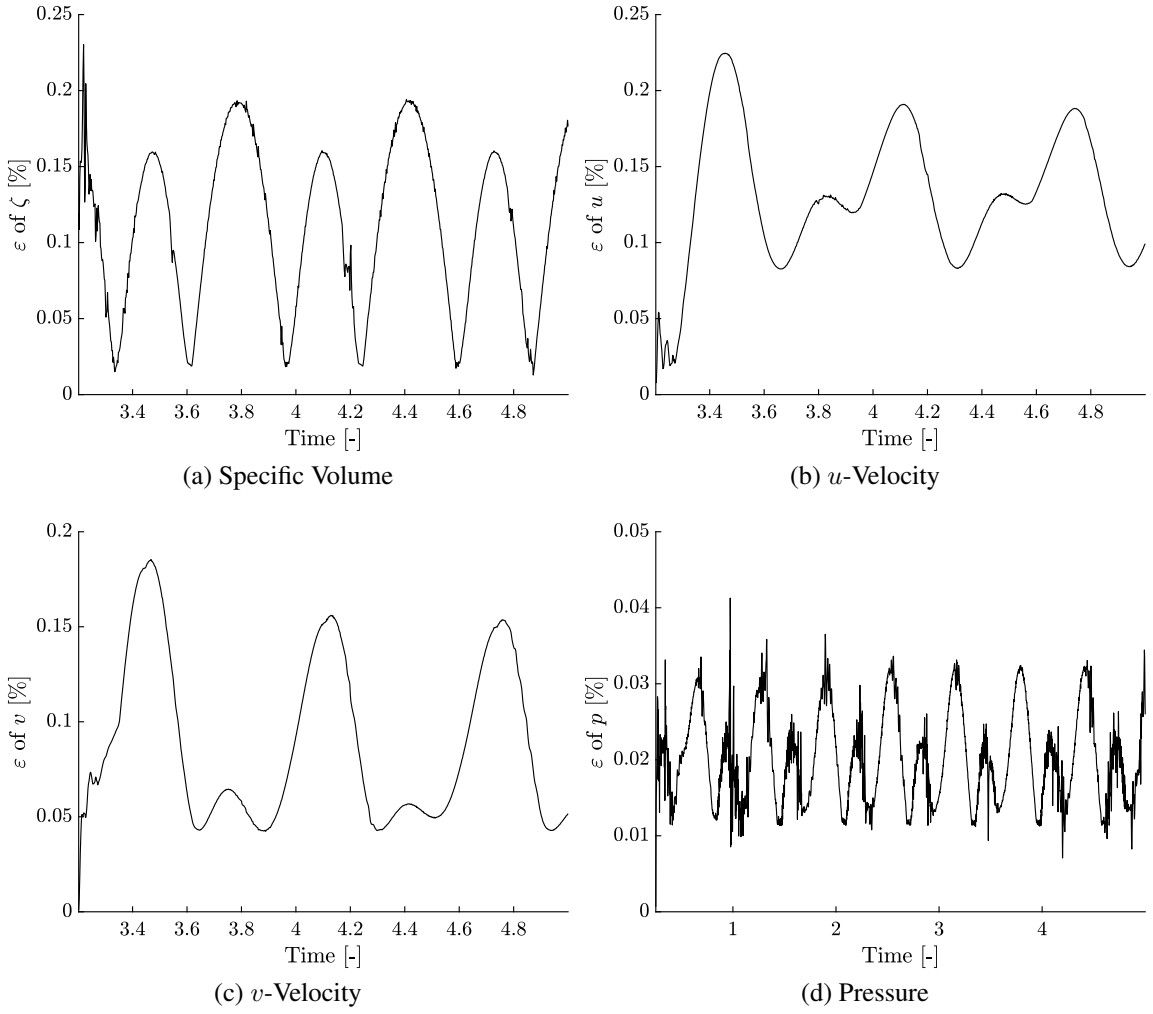
(b) $u$-Velocity

(c) $v$-Velocity

(d) Pressure

Figure 5.9: Time-averaged relative error between the ROM solution and the FOM solution for a two-dimensional channel flow with outlet pressure oscillation of angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$.

0.11% for all variables. To assess the variation of the error in time, Fig. 5.10 shows the maximum relative error across all nodes versus time for case 1. The maximum relative error across all nodes shows an oscillatory motion, but remains small for all state variables. To summarize the results of




(a) Specific Volume

(b) $u$-Velocity

(c) $v$-Velocity

(d) Pressure

Figure 5.10: Two-dimensional channel flow with an outlet pressure oscillation with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$: time variation of maximum relative error between the ROM and FOM solutions across all nodes.

all cases, Table 5.5 shows the maximum spatial and temporal relative error for all three validation cases. All cases match well with the FOM solution, the maximum relative error being less than 0.5%.

| Case | $\max_{x,t}(\epsilon)$ (%) $\zeta$ | $u$ | $v$ | $p$ |
|---|---|---|---|---|
| 1 | 0.20 | 0.19 | 0.19 | 0.04 |
| 2 | 0.40 | 0.29 | 0.27 | 0.09 |
| 3 | 0.23 | 0.21 | 0.13 | 0.07 |

Table 5.5: Two-dimensional channel flow: maximum spatial and temporal relative error between the ROM and FOM, for all on-reference cases.

### 5.3 Axisymmetric Nozzle[2]

This section presents results for inviscid flow through an axisymmetric nozzle. The nozzle is a converging-diverging nozzle with area variation given by [74].

$$A(x) = \begin{cases} 1.75 - 0.75\cos\left(\pi\left(2x/\ell - 1\right)\right) & 0 \leq x \leq \ell/2 \\ 1.25 - 0.25\cos\left(\pi\left(2x/\ell - 1\right)\right) & \ell/2 \leq x \leq \ell \end{cases}$$

The length of the nozzle $\ell$ was 254 mm (10 inches) while the throat was located at 127 mm (5 inches) downstream from the inlet. The inlet stagnation pressure and temperature were 104,191 Pa and 290.435 K, respectively. The flow entered the inlet along the $x$-axis. The mean outlet pressure was 102,107 Pa.

The FOM was used to simulate the steady flow in the axisymmetric nozzle for three grids: coarse, medium and fine with 3045, 22,673, and 175,041 grid points, respectively. Table 5.6 summarizes the information on the three grids. Figure 5.11 shows the medium grid for the axisymmetric nozzle.

Table 5.7 shows the grid convergence of the steady-state Mach number at the outlet. As the grid was refined, the variation of Mach number decreased. The percent change between grid refinements is also shown in Table 5.7. Since the variation between the medium and fine grid was small, results for the medium grid are shown herein.

[2]This section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

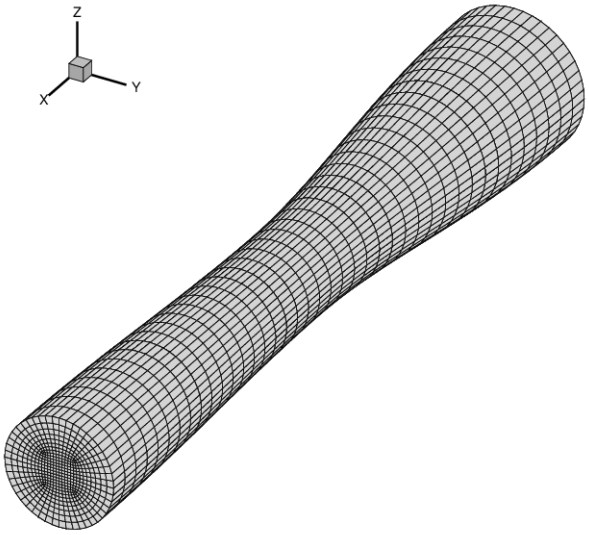



Figure 5.11: Computational mesh of the axisymmetric nozzle with 22,673 nodes.

| Type | Nodes in $x$-direction | Nodes in circumferential direction | Nodes in $yz$-plane | Total number of nodes, $N$ |
|---|---|---|---|---|
| Coarse | 21 | 24 | 145 | 3045 |
| Medium | 41 | 48 | 553 | 22,673 |
| Fine | 81 | 96 | 2161 | 175,041 |

Table 5.6: Grids refinement of the axisymmetric nozzle.

| Grid | Steady-State Outlet Mach Number | Change from Previous (%) |
|---|---|---|
| Coarse | 0.1552 | - |
| Medium | 0.1600 | 3.12 |
| Fine | 0.1644 | 2.74 |

Table 5.7: Grid convergence of the axisymmetric nozzle.

The nozzle flow was made unsteady in two ways: (i) through an oscillating outlet pressure boundary condition (5.4), and (ii) through an oscillating wall deformation. The results of both cases are given for both on- and off-reference conditions in the following sections.

### 5.3.1 Oscillating Outlet Pressure

An outlet pressure oscillation boundary condition (5.4) was imposed onto the axisymmetric nozzle. Table 5.8 shows the five cases created for validating the ROM by varying the angular velocity and amplitude of the outlet pressure oscillation. Cases one to three are listed as on-reference cases,

| Case | $\tilde{\omega}$ [rad/sec] | $\tilde{A}$ | Type |
|---|---|---|---|
| 1 | 10 | 0.010 | on-reference |
| 2 | 10 | 0.020 | on-reference |
| 3 | 20 | 0.010 | on-reference |
| 4 | 10 | 0.015 | off-reference |
| 5 | 15 | 0.010 | off-reference |

Table 5.8: On- and off-reference cases for the axisymmetric nozzle with an oscillating outlet pressure.

that is, cases for which the zeta-variable ROM reproduces a FOM solution. Cases four and five are listed as off-reference cases, that is, cases for which the zeta-variable ROM will predict a FOM solution. The FOM generated snapshots for 3 periods, each period consisting of approximately 666 time steps. A total of 2000 snapshots were used to generate the autocorrelation matrix $\mathbf{R}$.

For all cases, three procedures were used to stabilize the ROM solution. The penalty method was imposed onto the ROM by enforcing a prescribed boundary condition at the outlet

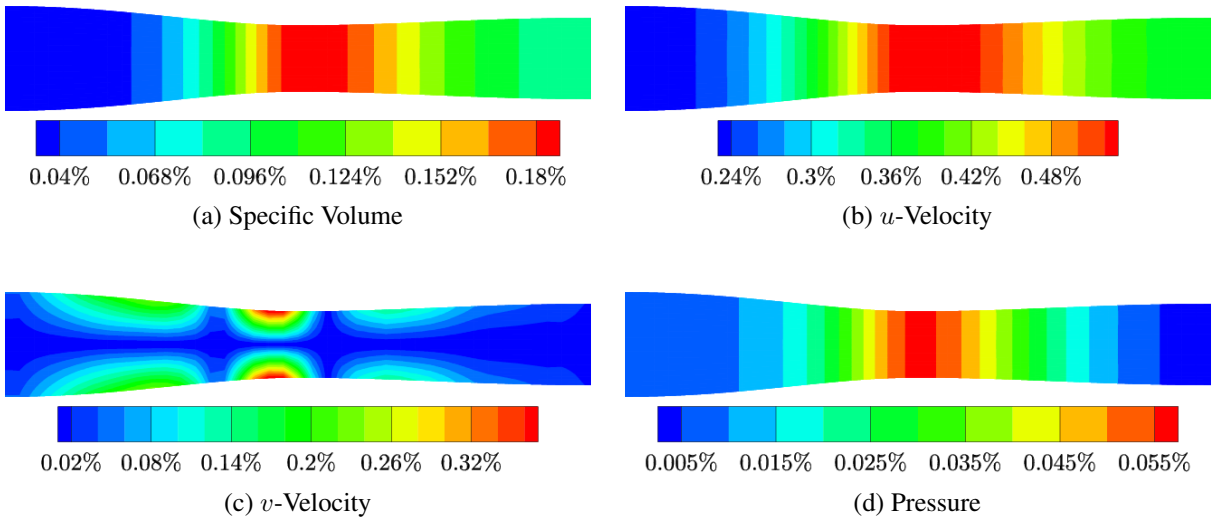
$$\mathbf{F}(t) = \left( (\gamma \mathbf{f}(t))^{\frac{1}{\gamma}}, 0, 0, 0, \mathbf{f}(t) \right)^T$$

where $\mathbf{f}(t) \in \mathbb{R}^{N_{\text{out}}}$ and $\mathbf{f}_i(t) = f(t)$ of (5.4), $1 \leq i \leq N_{\text{out}}$. The penalty parameter $\boldsymbol{\tau}$ was solved for using (4.8). Artificial dissipation was used to obtain a stable solution (4.10).

On-reference results are shown only for case 1 since all cases 2 and 3 showed similar results. Cases 1 to 3 used $n^{\zeta} = n^p = 1, n^u = 3$, and $n^v = n^w = 2$ modes with a modified number of modes of $n^u = 2$ in the $y$- and $z$-momentum equations and in the conservation of energy equation. Case

1 had values for the artificial dissipation parameter of $\tilde{\nu} = (2, 0.45, 4, 4, 2)^T \times 10^{-4}$. These values were found empirically, by increasing the amount of dissipation until the ODEs became stable.

Equations (5.5) and (5.6) were used to find the relative error between the ROM and the FOM. Figure 5.12 shows the time-averaged relative error between the ROM and the FOM solutions. The comparison for $z$-velocity is not shown since it is identical to $y$-velocity for an axisymmetric nozzle. The ROM solution for $x$-velocity used only one mode for its reconstruction (2.1), while three modes were needed while solving for stability. The ROM and FOM solutions were in good agreement,




(a) Specific Volume (b) $u$-Velocity

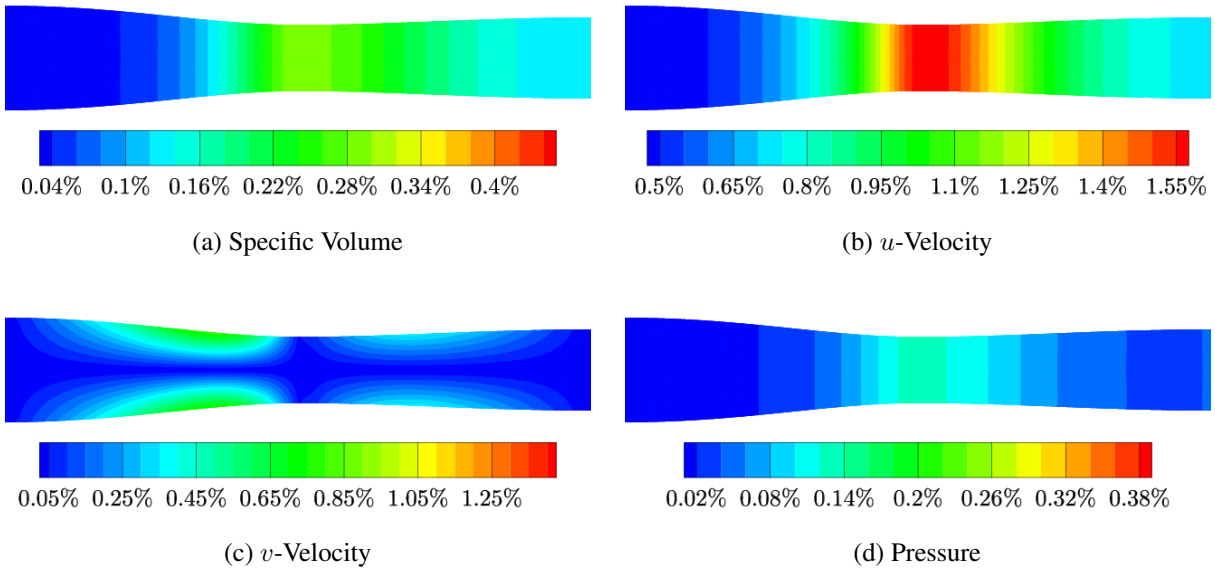
(c) $v$-Velocity (d) Pressure

Figure 5.12: Time-averaged relative error between the ROM and FOM solutions for an axisymmetric nozzle with outlet pressure oscillation of angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.01$.

with a time-averaged relative error below 1% for all state variables. All three stabilizing methods were necessary to obtain a converged solution.

Off-reference results are shown for both cases 4 and 5. To develop the basis functions for case 4, the snapshots of cases 1 and 2 were combined to produce the autocorrelation matrix. The basis functions for case 5 were generated using the snapshots of cases 1 and 3. Artificial dissipation was applied to both cases to limit the growth of spurious modes. The artificial dissipation parameters were given the upper bound value of (4.11).

Figure 5.13 shows the time-averaged relative error of the ROM solution with respect to the FOM solution for case 4. The FOM solution was only generated for the purpose of comparison,




(a) Specific Volume

(b) $u$-Velocity

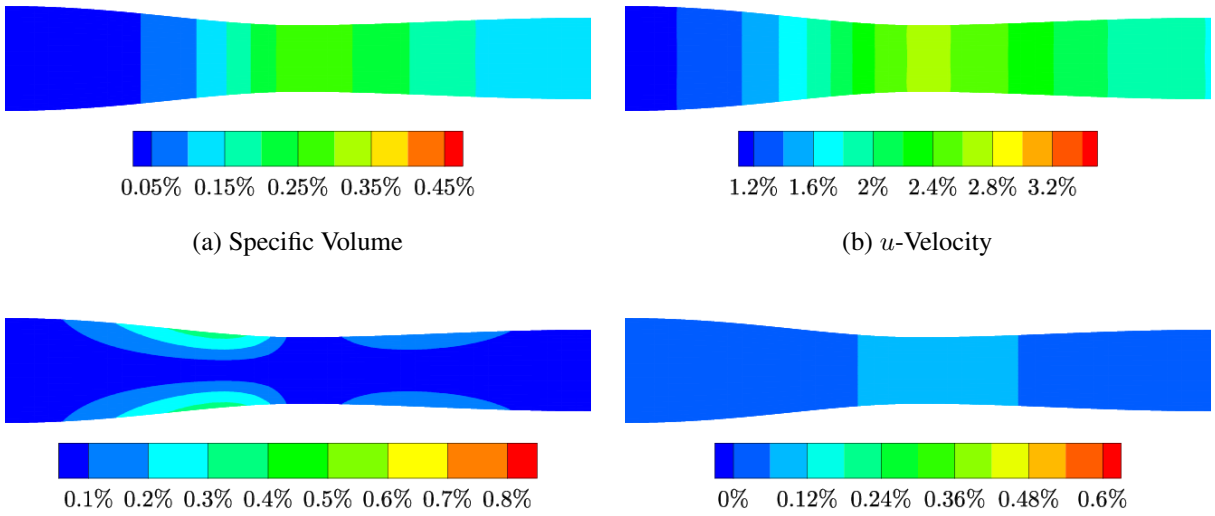
(c) $v$-Velocity

(d) Pressure

Figure 5.13: Time-averaged relative error between the ROM off-reference solution and the FOM solution for an axisymmetric nozzle with outlet pressure oscillation with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.015$.

and its snapshots were not used in the development of the autocorrelation matrix. For this case, artificial dissipation was only applied to the velocity components. The ROM used the following number of modes: $n^{\zeta} = n^{p} = 1$ and $n^{u} = n^{v} = n^{w} = 3$. There was good agreement between the time-averaged ROM and FOM solutions. The largest error occurred in the $x$-velocity and was less than 1.6%.

Figure 5.14 shows the time-averaged relative error of the ROM solution with respect to the FOM solution for case 5. The ROM used the following number of modes: $n^{\zeta} = n^{p} = 1$ and $n^{u} = n^{v} = n^{w} = 2$. Similarly to case 4, artificial dissipation was only applied to the velocity components. The ROM solution agreed well with the FOM solution, the largest error being in $x$-velocity.




(a) Specific Volume

(b) $u$-Velocity

(c) $v$-Velocity

(d) Pressure

Figure 5.14: Time-averaged relative error between the ROM off-reference solution and the FOM solution for an axisymmetric nozzle with outlet pressure oscillation with angular velocity $\tilde{\omega} = 15$ rad/sec and amplitude $\tilde{A} = 0.01$.

Table 5.9 summarizes the maximum spatial and temporal relative error for both the on- and off-reference cases. The off-reference results have a larger error than the on-reference results with

| | $\max_{x,t}(\epsilon)$ (%) | | | | |
|---|---|---|---|---|---|
| Case | $\zeta$ | $u$ | $v$ | $w$ | $p$ |
| 1 | 0.29 | 0.70 | 0.48 | 0.48 | 0.13 |
| 2 | 0.49 | 0.56 | 0.64 | 0.25 | 0.62 |
| 3 | 0.34 | 0.69 | 0.68 | 0.29 | 0.38 |
| 4 | 0.92 | 4.32 | 1.49 | 1.44 | 1.24 |
| 5 | 0.64 | 4.96 | 1.00 | 0.62 | 0.91 |

Table 5.9: Axisymmetric nozzle with outlet pressure oscillation: maximum spatial and temporal relative error between the ROM and FOM solutions, for all cases.

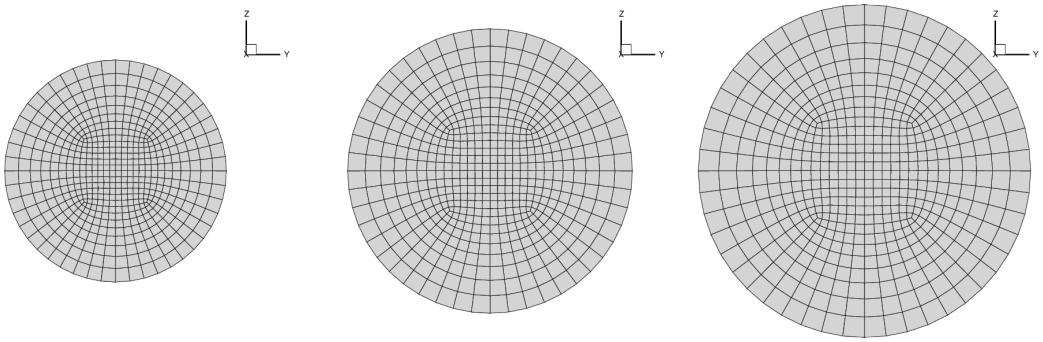
the largest error occurring in $x$-velocity, which was less than 5% for both off-reference cases. All other variables had errors less than 2% for both on- and off-reference results.

### 5.3.2 Deforming Boundary

An oscillating wall boundary deformation was imposed on the axisymmetric nozzle by varying in time the radius of the nozzle at every *x* location according to

$$r_{\text{forced}}(t) = \bar{r}\left(1 + \tilde{A}\sin\left(2\pi\tilde{\omega}t\right)\right) \tag{5.7}$$

where $r_{\text{forced}}$ is the radius of the nozzle that is deformed, $\bar{r}$ is the nozzle undeformed radius, $\tilde{A}$ is the amplitude of the deformation, and $\tilde{\omega}$ is the angular velocity of the deformation. An example of the deformation for an amplitude $\tilde{A} = 0.2$ is shown in Figure 5.15.



(a) Minimum, $t = -\frac{1}{4\tilde{\omega}}$ (b) Undeformed, $t = 0$ (c) Maximum, $t = \frac{1}{4\tilde{\omega}}$

Figure 5.15: Nozzle inlet deformation for amplitude $\tilde{A} = 0.2$.

To validate the zeta-variable POD-based ROM, five cases were investigated by varying the amplitude and angular velocity of the nozzle deformation. Table 5.10 summarizes these cases. A back pressure of 102,100 Pa was held constant at downstream infinity during the deformation. The FOM solutions were generated for three periods, each period consisting of 100 snapshots for a total of 300 snapshots to generate the autocorrelation matrix **R**. The zeta-variable ROM used the following number of nodes for all cases: $n^{\zeta} = n^{u} = n^{v} = n^{w} = n^{p} = 1$.

To stabilize the zeta-variable ROM and preserve accuracy, the penalty method was applied to each state variable. To apply it to each state variable, a prescribed boundary condition needed to be

| Case | $\tilde{\omega}$ [rad/sec] | $\tilde{A}$ | Type |
|---|---|---|---|
| 1 | 10 | 0.2 | on-reference |
| 2 | 10 | 0.4 | on-reference |
| 3 | 20 | 0.2 | on-reference |
| 4 | 10 | 0.3 | off-reference |
| 5 | 15 | 0.2 | off-reference |

Table 5.10: On- and off-reference cases for the axisymmetric nozzle with an oscillating boundary deformation.

defined for each state variable. To define the prescribed boundary condition, several approximate parameters were defined. The spatial and temporal average of the state vector across a prescribed boundary of the FOM is

$$\hat{Z}_k = \left\langle \bar{Z}_k(\mathbf{x}_{\text{BC}}) \right\rangle \tag{5.8}$$

where $\bar{\cdot}$ denotes a spatial average, $\langle\cdot\rangle$ denotes a time average, and $\mathbf{x}_{\text{BC}}$ are the nodes along the prescribed boundary. Recall that the zeta-variable state vector is $\mathbf{Z} = (\zeta, u, v, w, p)^T$, such that $Z_1 = \zeta$, $Z_2 = u$, *etc.* The amplitude is defined using the maximum and minimum values over time across a prescribed boundary

$$\hat{A}_k = \frac{\max\left(\bar{Z}_k(\mathbf{x}_{\text{BC}})\right) - \min\left(\bar{Z}_k(\mathbf{x}_{\text{BC}})\right)}{2\hat{Z}_k}. \tag{5.9}$$

Using definitions (5.14) and (5.9), an approximation of the solution at the prescribed boundary is defined

$$Z_{k_{\text{approx}}} = \hat{Z}_k \left(1 + \hat{A}_k \sin\left(\tilde{\omega}t - \hat{\varphi}_k\right)\right), \; k \in [1, D+2] \tag{5.10}$$

where the value of the phase $\hat{\varphi}_k$ is determined such that the peak values of the FOM solution and its approximation are in phase, that is, $\tilde{\omega}t^{\text{peak}} - \hat{\varphi}_k = \tilde{\omega}t^{\text{peak}}_{\text{FOM}}$. Consequently,

$$\hat{\varphi}_k = \tilde{\omega}\left(t^{\text{peak}} - t^{\text{peak}}_{\text{FOM}}\right). \tag{5.11}$$

The penalty method was imposed onto the ROM using these approximate equations, where the prescribed boundary condition was applied at the outlet

$$\mathbf{F}(t) = \left(\mathbf{Z}_{\zeta_{\text{approx}}}, \mathbf{Z}_{u_{\text{approx}}}, \mathbf{Z}_{v_{\text{approx}}}, \mathbf{Z}_{w_{\text{approx}}}, \mathbf{Z}_{p_{\text{approx}}}\right)^T$$

each component being a vector of size $N_{\text{out}}$. The penalty parameter $\boldsymbol{\tau}$ was calculated for each variable by solving (4.8).

The spatial and temporal averages $\hat{Z}_k$, $\hat{A}_k$ and $\hat{\varphi}_k$ were calculated based on the FOM solutions for the on-reference cases and are shown in Table 5.11. One can note that $\hat{Z}_k$ remains approximately

| | Case | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| $\hat{\zeta}$ | 1.00 | 1.00 | 1.00 |
| $\hat{A}_\zeta \times 10^{-5}$ | 1.52 | 2.93 | 1.73 |
| $\hat{\varphi}_\zeta$ | -0.82 | -0.69 | -0.75 |
| $\hat{u}$ | 0.16 | 0.16 | 0.16 |
| $\hat{A}_u \times 10^{-3}$ | 0.81 | 1.68 | 0.89 |
| $\hat{\varphi}_u$ | 3.64 | 3.77 | -1.26 |
| $\hat{v} \times 10^{-6}$ | -5.11 | -5.39 | -5.16 |
| $\hat{A}_v$ | -0.34 | -0.65 | -0.40 |
| $\hat{\varphi}_v$ | 9.36 | 28.27 | -28.27 |
| $\hat{w} \times 10^{-6}$ | -5.94 | -6.06 | -5.95 |
| $\hat{A}_w$ | -0.27 | -0.54 | -0.30 |
| $\hat{\varphi}_w$ | 15.77 | 15.77 | -9.42 |
| $\hat{p}$ | 0.72 | 0.72 | 0.72 |
| $\hat{A}_p \times 10^{-5}$ | 0.97 | 1.74 | 0.72 |
| $\hat{\varphi}_p$ | 9.36 | 9.42 | 28.40 |

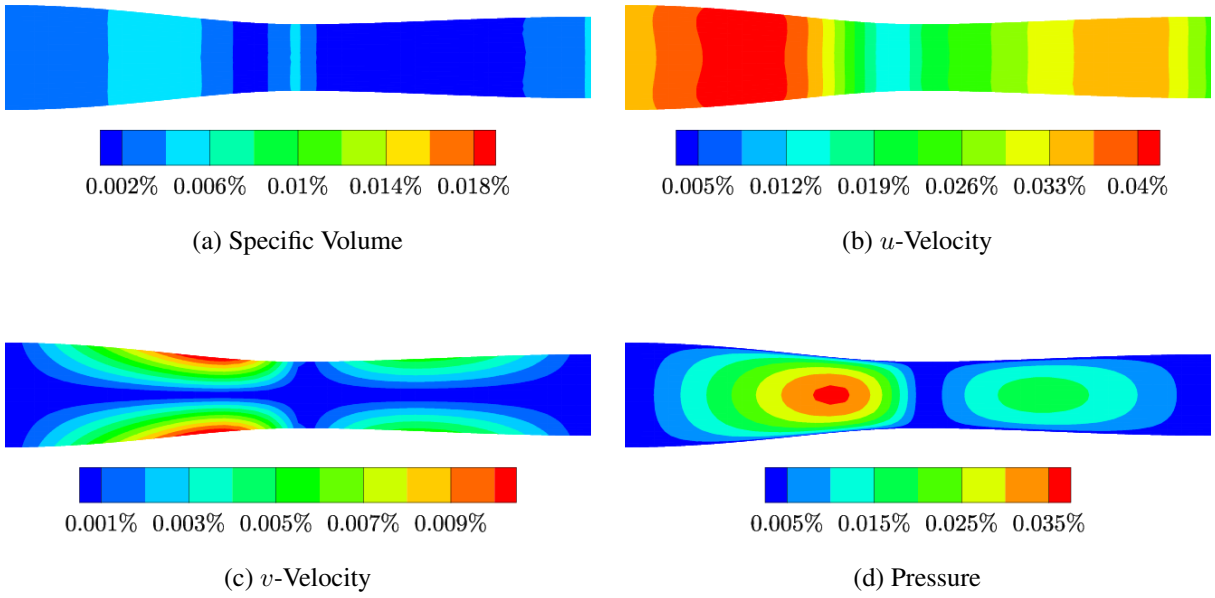
Table 5.11: Mean values of state variables, amplitudes and phase angles for a nozzle deformation of the axisymmetric nozzle for all on-reference cases.

constant for all cases while the amplitude $\hat{A}_k$ scales approximately with the actual amplitudes $\tilde{A}$. For example, case 2 has an amplitude, $\tilde{A} = 0.4$, that is twice as large as the amplitude for case

1, $\tilde{A} = 0.2$. Similarly, using conservation of mass as an example, the approximate amplitude of case 2, $\hat{A}_\zeta = 2.93 \times 10^{-5}$, is approximately twice as large as the approximate amplitude for case 1, $\hat{A}_\zeta = 1.52 \times 10^{-5}$.

It is important to note that with the nozzle deformation imposed, the coefficients of the ODEs of the ROM that contain derivative terms will become time-dependent in the spatial term. If the deformation is small, one can assume the coefficients remain approximately constant in time, using the coefficients calculated at the mean deformation. In this way, the coefficients of the ODEs remain time-independent. This assumption was applied to this case.

Only case 1 results are shown here since cases 2 and 3 showed similar results. As before, the errors were found using (5.5) and (5.6). Figure 5.16 shows the time-averaged relative error between the ROM and FOM solutions. The $z$-velocity results are not shown since they were identical to




(a) Specific Volume

(b) $u$-Velocity

(c) $v$-Velocity

(d) Pressure

Figure 5.16: Time-averaged relative error between the ROM and FOM solutions for an axisymmetric nozzle deforming with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.2$.

those of the $y$-velocity. The time-averaged relative errors showed a good agreement between the

ROM and the FOM solutions.

Off-reference results for cases 4 and 5 are shown here. Grassmann interpolation was used to find the interpolated basis for cases 4 and 5. Cases 1 and 2 were used to develop the interpolated basis functions for case 4. Cases 1 and 3 were used to find the interpolated basis functions for case 5.

The penalty method was applied at the outlet using an outlet approximation (5.13) for each variable as the prescribed boundary condition, similarly to what was done for the on-reference cases. Unlike the on-reference cases, the approximate values, which were found from the FOM solution, cannot be directly computed using (5.14), (5.9), and (5.16). Therefore, the approximate values, $\hat{q}_k$, $\hat{A}_k$, and $\hat{\varphi}_k$, are found by interpolating from the on-reference solutions.

As noted, $\hat{Z}_k$ are almost independent of the case number, while the amplitudes $\hat{A}_k$ scale with the actual amplitudes $\tilde{A}_k$. Table 5.12 shows the mean values $\hat{Z}_k$, $\hat{A}_k$, and $\hat{\varphi}_k$ for cases 4 and 5. The table includes "exact" values, that is, derived directly from respective FOM solutions evaluated at the amplitudes and angular velocities of cases 4 and 5, and values scaled from case 1. The values scaled from case 1 matched well the "exact" values. The phase angle for case 4 was determined by linear interpolation between cases 1 and 2. Similarly, the phase angle for case 5 was determined by linear interpolation between cases 1 and 3.

Figures 5.17 and 5.18 show the time-averaged relative errors of the ROM solution with respect to the FOM solution for cases 4 and 5, respectively. In both cases, the off-reference ROM solution matched well the FOM solution, the time-averaged relative error being less than 0.1%.

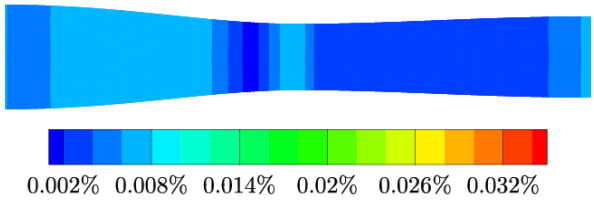
Table 5.13 shows the maximum spatial and temporal relative error for all cases. The maximum relative error between the off-reference ROM and the FOM solutions was less than 1% while the maximum relative error was less than 0.5% for the on-reference cases.

| | Case | | | |
|---|---|---|---|---|
| | Exact (FOM) | | Scaled | |
| | 4 | 5 | 4 | 5 |
| $\hat{\zeta}$ | 1.00 | 1.00 | 1.00 | 1.00 |
| $\hat{A}_\zeta \times 10^{-5}$ | 2.26 | 1.70 | 2.28 | 1.52 |
| $\hat{\varphi}_\zeta$ | 11.81 | 17.63 | -0.75 | -0.90 |
| $\hat{u}$ | 0.16 | 0.16 | 0.16 | 0.16 |
| $\hat{A}_u \times 10^{-3}$ | 1.23 | 0.66 | 1.21 | 0.81 |
| $\hat{\varphi}_u$ | -2.58 | 41.54 | 3.71 | 2.26 |
| $\hat{v} \times 10^{-6}$ | -5.05 | -5.05 | -5.11 | -5.11 |
| $\hat{A}_v$ | -0.52 | -0.38 | -0.51 | -0.34 |
| $\hat{\varphi}_v$ | 3.08 | 40.42 | 18.82 | -3.58 |
| $\hat{w} \times 10^{-6}$ | -5.82 | -5.87 | -5.94 | -5.94 |
| $\hat{A}_w$ | -0.42 | -0.30 | -0.41 | -0.27 |
| $\hat{\varphi}_w$ | 22.05 | 27.95 | 15.77 | 8.29 |
| $\hat{p}$ | 0.72 | 0.72 | 0.72 | 0.72 |
| $\hat{A}_p \times 10^{-5}$ | 1.41 | 0.96 | 1.46 | 0.97 |
| $\hat{\varphi}_p$ | 21.99 | 9.10 | 9.39 | 17.67 |

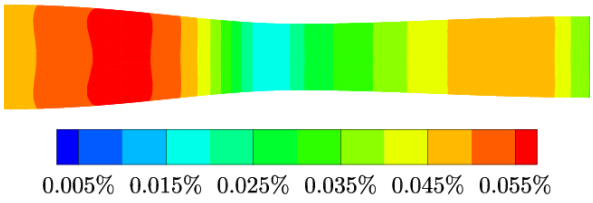
Table 5.12: Axisymmetric nozzle with oscillating boundary deformation: exact and scaled mean values of state variables, amplitudes and phase angles for all off-reference cases.

| | $\max_{x,t}(\epsilon)$ (%) | | | | |
|---|---|---|---|---|---|
| Case | $\zeta$ | $u$ | $v$ | $w$ | $p$ |
| 1 | 0.04 | 0.08 | 0.01 | 0.04 | 0.02 |
| 2 | 0.06 | 0.17 | 0.05 | 0.09 | 0.03 |
| 3 | 0.06 | 0.42 | 0.06 | 0.36 | 0.15 |
| 4 | 0.05 | 0.12 | 0.96 | 0.48 | 0.06 |
| 5 | 0.04 | 0.24 | 0.48 | 0.31 | 0.11 |

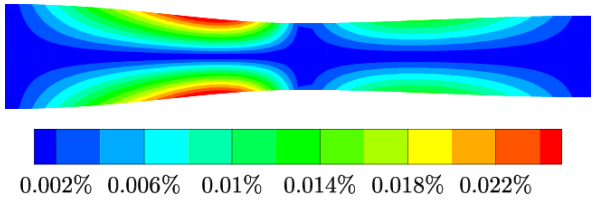
Table 5.13: Axisymmetric nozzle with oscillating walls: maximum spatial and temporal relative error between the ROM and FOM solutions.



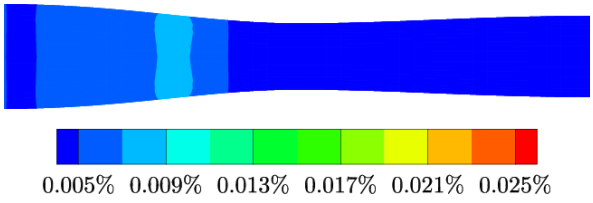

(a) Specific Volume




(b) $u$-Velocity

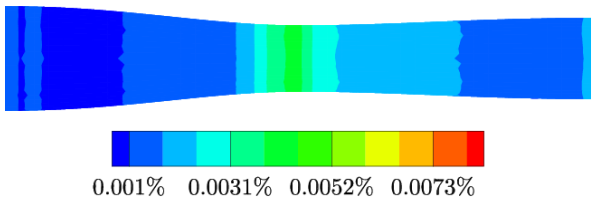



(c) $v$-Velocity

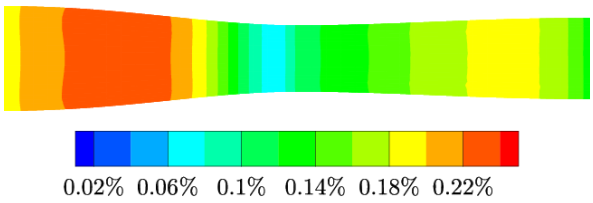



(d) Pressure

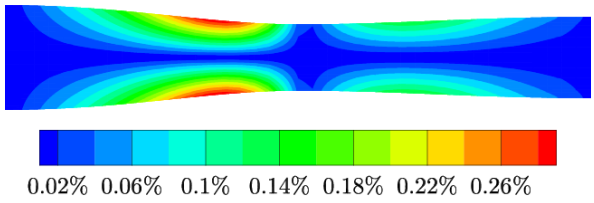
Figure 5.17: Time-averaged relative error between the ROM off-reference and FOM solutions for an axisymmetric nozzle with walls oscillating with angular velocity $\tilde{\omega} = 10$ rad/sec and amplitude $\tilde{A} = 0.3$.



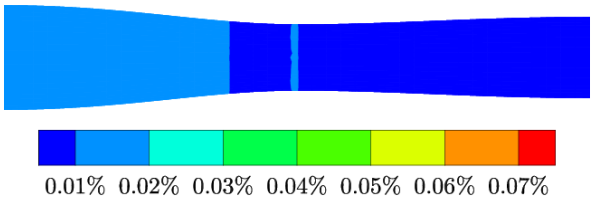

(a) Specific Volume




(b) $u$-Velocity




(c) $v$-Velocity

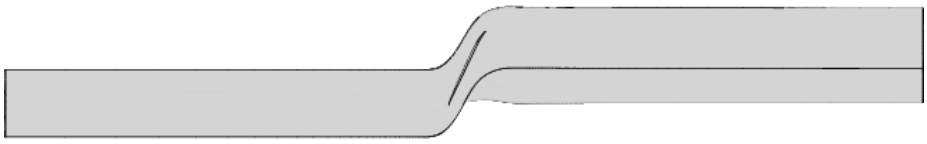



(d) Pressure

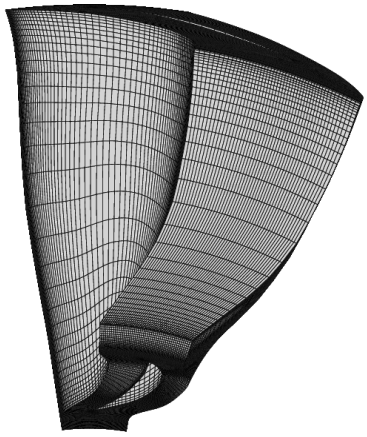
Figure 5.18: Time-averaged relative error between the ROM off-reference and FOM solutions for an axisymmetric nozzle with walls oscillating with angular velocity $\tilde{\omega} = 15$ rad/sec and amplitude $\tilde{A} = 0.2$.

## 5.4 Rotor 67

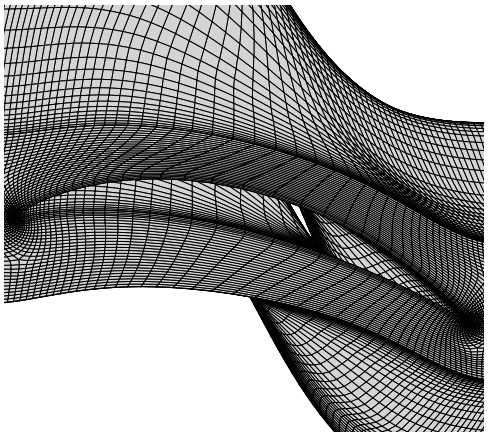
This section presents results of inviscid flow through a twenty-two-blade first-stage rotor of a two-stage fan, referred to as Rotor 67. Rotor 67 was developed and tested at NASA Lewis [75]. The inlet stagnation pressure and temperature are 124,453 Pa and 305.74 K, respectively. The flow enters the inlet along the x-axis. The mean outlet pressure at the hub is 136,898 Pa. A FOM solution for this case was generated and validated against the experimental data [76]. This FOM solved the Euler equations written in a rotating reference frame about the $x$-axis (3.7). Figure 5.19 shows the computational mesh of Rotor 67 that used $299,844$ grid nodes.



(a) Computational domain



(b) Detail of hub-to-tip mesh



(c) Detail of mesh near airfoil

Figure 5.19: NASA Rotor 67: computational mesh with $N = 299,844$ grid nodes.

The inlet and outlet boundaries were located approximately 5 chords away from the airfoil to reduce spurious boundary reflections. Periodic boundary conditions were used to reduce the

computational domain to one blade passage. The pressure variation from hub to tip across the outlet was solved using the axisymmetric radial momentum equation

$$\frac{\partial p}{\partial r} = \frac{\rho v_\theta^2}{r} \tag{5.12}$$

where $v_\theta = (vz - wy)/r$. The angular velocity about the *x*-axis was $\omega_x$=16,520 RPM. Integrating (5.12) from the hub to a radial location $r$, $r \leq r_{\text{tip}}$, yields

$$p(r) - p(r_{\text{hub}}) = \int_{r_{\text{hub}}}^{r} \frac{\rho v_\theta^2}{r} dr.$$

The unsteadiness was introduced in the FOM through an oscillating outlet pressure at the hub $p(r_{\text{hub}})$ that followed the form of Eq. (5.4). Three amplitudes $\tilde{A}$ and three angular velocities $\tilde{\omega}$ were considered, as shown in Table 5.14. Similar to the previous results, three on-reference solutions and two off-reference solutions were developed. The FOM generated snapshots for five periods.

| Case | $\tilde{\omega}$ [rad/sec] | $\tilde{A}$ | Type |
|---|---|---|---|
| 1 | 38,059 | 0.050 | on-reference |
| 2 | 38,059 | 0.100 | on-reference |
| 3 | 37,582 | 0.050 | on-reference |
| 4 | 38,059 | 0.075 | off-reference |
| 5 | 37,765 | 0.050 | off-reference |

Table 5.14: NASA Rotor 67: on- and off-reference cases for oscillating outlet pressure.

Fifty snapshots were saved for each period. All cases had a pressure $\bar{p}$=136,900 Pa and a phase shift $\varphi = 0$.

For $\mathbf{x}_{\text{BC}}$ denoting the nodes along the prescribed boundary, the penalty method was imposed onto the ROM for each state variable using an approximation of the boundary solution

$$Z_{k_{\text{approx}}} = \hat{Z}_k \left(1 + \hat{A}_k \sin\left(\tilde{\omega} t - \hat{\varphi}_k\right)\right), \; k \in [1, D+2]. \tag{5.13}$$

Here, $\hat{Z}_k$ is the spatial and temporal average of the state vector

$$\hat{Z}_k = \left\langle \bar{Z}_k(\mathbf{x}_{\text{BC}}) \right\rangle \tag{5.14}$$

where $\bar{\cdot}$ denotes a spatial average. The approximate amplitude, $\hat{A}_k$, is found by finding the average of the maximum and minimum in time of $\hat{Z}_k$.

$$\hat{A}_k = \frac{\max\left(\bar{Z}_k(\mathbf{x}_{\text{BC}})\right) - \min\left(\bar{Z}_k(\mathbf{x}_{\text{BC}})\right)}{2\hat{Z}_k} \tag{5.15}$$

The approximate phase, $\hat{\varphi}_k$, is found from the peak values of the FOM solution.

$$\hat{\varphi}_k = \tilde{\omega}\left(t^{\text{peak}} - t^{\text{peak}}_{\text{FOM}}\right) \tag{5.16}$$

The final prescribed boundary condition was

$$\mathbf{F}(t) = \left((\gamma \mathbf{f}(t))^{\frac{1}{\gamma}}, \mathbf{Z}_{u_{\text{approx}}}, \mathbf{Z}_{v_{\text{approx}}}, \mathbf{Z}_{w_{\text{approx}}}, \mathbf{Z}_{p_{\text{approx}}}\right)^T$$

where $\mathbf{f}(t) \in \mathbb{R}^{N_{\text{out}}}$ and $\mathbf{Z}_{k_{\text{approx}}} \in \mathbb{R}^{N_{\text{hub}}}, k \in [2, D+2]$, with $N_{\text{hub}}$ being the number of nodes along the outlet hub boundary. The penalty method was applied at the hub to avoid solving (5.12) from hub to tip at each time step in the ROM. An isentropic relation was used for determining the boundary condition for the specific volume [39].

The penalty parameter $\boldsymbol{\tau}$ was solved for using (4.7) for each state variable. The approximate values of (5.13) were determined from the FOM solution for each case using (5.14) to (5.16), and these values are shown in Table 5.15. The value for $\hat{Z}$ does not change between the cases while the value for $\hat{A}$ scales similarly to the actual amplitude values. For example, the amplitude ratio from case 1 to case 2 using the values in Table 5.14 is 2. The amplitude ratio from case 1 to case 2 using the values in Table 5.15 ranges from 2 to 2.01.

For conciseness and since cases 2 and 3 showed similar results, only case 1 results are shown

| | Case | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| $\hat{u}$ | 0.50 | 0.50 | 0.50 |
| $\hat{A}_u \times 10^{-2}$ | 0.95 | 1.91 | 0.97 |
| $\hat{\varphi}_u$ | -13.83 | -13.83 | -19.86 |
| $\hat{v}$ | -0.67 | -0.67 | -0.67 |
| $\hat{A}_v \times 10^{-4}$ | -2.22 | -4.47 | -2.27 |
| $\hat{\varphi}_v$ | 4.53 | 4.65 | -1.49 |
| $\hat{w}$ | 0.12 | 0.12 | 0.12 |
| $\hat{A}_w \times 10^{-2}$ | 0.73 | 1.46 | 0.74 |
| $\hat{\varphi}_w$ | -19.99 | -19.99 | -19.86 |
| $\hat{p}$ | 1.08 | 1.08 | 1.08 |
| $\hat{A}_p \times 10^{-2}$ | 0.52 | 1.04 | 0.52 |
| $\hat{\varphi}_p$ | -23.13 | -23.13 | 2.23 |

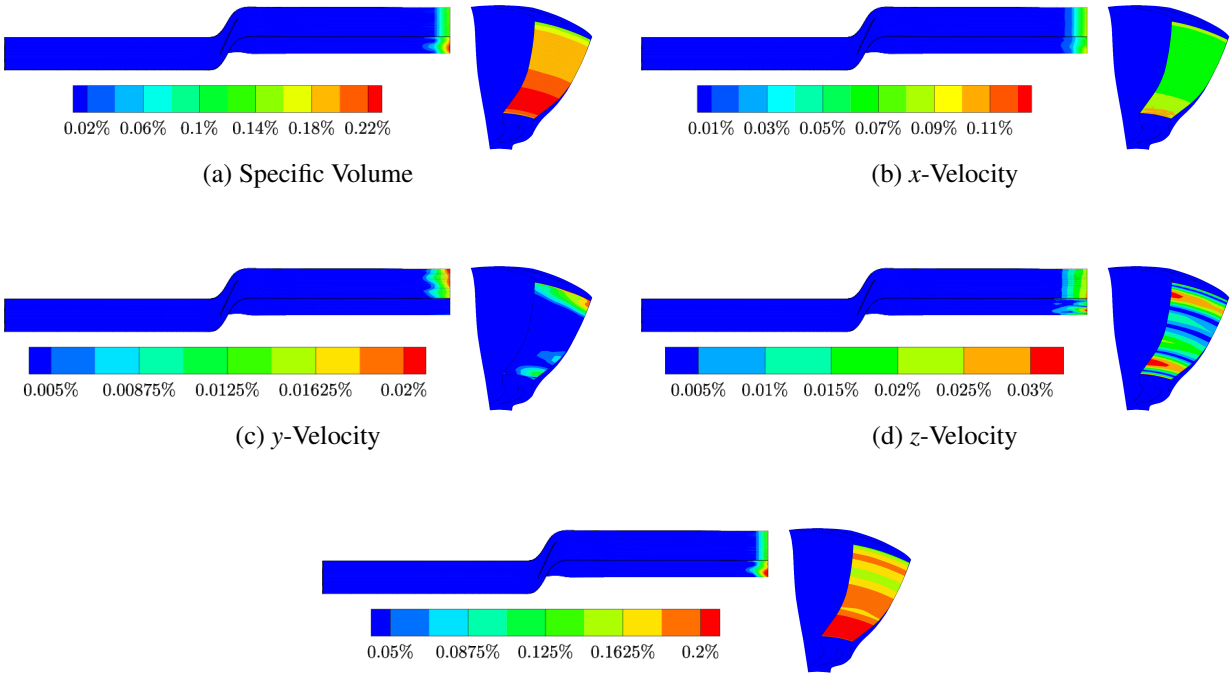
Table 5.15: NASA Rotor 67 with outlet pressure oscillation: mean values of state variables, amplitudes and phase angles.

here for the on-reference conditions. Cases 1 to 3 were solved using $n^{\zeta} = n^{u} = n^{v} = n^{w} = n^{p} = 1$.

An error between the ROM and the FOM was calculated using (5.5) and (5.6). Figure 5.20 shows the time-averaged relative error between the FOM and ROM solutions. The time-averaged relative error is less than 0.5% for all state variables. The largest errors occur at the outlet, where the penalty-imposed boundary conditions were applied. Of note in this case is that there exists a shock wave near the leading edge of the blade. The error values are not increased at the shock location, therefore the zeta-variable ROM solution accurately captures the flow discontinuity from the FOM solution. Table 5.16 summarizes the results of all on-reference cases for Rotor 67. Each case shows good agreement between the zeta-variable ROM solution to its respective FOM solution with a maximum relative error over all time and space of less than 1%.

The Grassmann interpolation was used to interpolate the basis functions to a new flow condition for the off-reference cases. Cases 1 and 2 were used to find the interpolated basis of case 4. Cases 1 and 3 were used to find the interpolated basis of case 5.

The penalty method was imposed similarly to the way it was applied to the on-reference cases.




(a) Specific Volume

(b) $x$-Velocity

(c) $y$-Velocity

(d) $z$-Velocity

(e) Pressure

Figure 5.20: Time-Averaged Relative Error of the ROM to the FOM for Rotor 67 with an outlet pressure oscillation with angular velocity $\tilde{\omega} = 38,059$ rad/sec and amplitude $\tilde{A} = 0.05$.

| | $\max_{x,t}(\epsilon)$ (%) | | | | |
|---|---|---|---|---|---|
| Case | $\zeta$ | $u$ | $v$ | $w$ | $p$ |
| 1 | 0.50 | 0.19 | 0.04 | 0.05 | 0.39 |
| 2 | 0.78 | 0.39 | 0.07 | 0.11 | 0.78 |
| 3 | 0.60 | 0.28 | 0.04 | 0.07 | 0.54 |

Table 5.16: NASA Rotor 67 with outlet pressure oscillation: maximum spatial and temporal relative errors between the ROM and FOM solutions, for all on-reference cases.

The approximate values of the amplitudes were scaled based on case 1 values given in Table 5.14. Table 5.17 shows the mean values $\hat{Z}_k$, $\hat{A}_k$ and $\hat{\varphi}_k$ for cases 4 and 5. The table includes "exact" values, that is, derived directly from respective FOM solutions evaluated at the amplitudes and angular velocities of cases 4 and 5, and values scaled from case 1. The values scaled from case 1 matched well with the "exact" values. The phase angle was interpolated using a linear interpolation:

the approximate values of the phase angles of case 4 were interpolated based on the phase angles of cases 1 and 2, and the phase angles of case 5 were interpolated based on the phase angles of cases 1 and 3.

| | Case | | | |
|---|---|---|---|---|
| | Exact (FOM) | | Scaled | |
| | 4 | 5 | 4 | 5 |
| $\hat{u}$ | 0.50 | 0.50 | 0.50 | 0.50 |
| $\hat{A}_u \times 10^{-2}$ | 1.43 | 0.96 | 1.43 | 0.97 |
| $\hat{\varphi}_u$ | -13.83 | 5.36 | -13.83 | -17.48 |
| $\hat{v}$ | -0.67 | -0.67 | -0.67 | -0.67 |
| $\hat{A}_v \times 10^{-4}$ | -3.34 | -2.25 | -3.33 | -2.27 |
| $\hat{\varphi}_v$ | 4.53 | -13.97 | 4.53 | 0.80 |
| $\hat{w} \times 10^{-6}$ | 0.12 | 0.12 | 0.12 | 0.12 |
| $\hat{A}_w \times 10^{-2}$ | 1.09 | 0.74 | 1.09 | 0.74 |
| $\hat{\varphi}_w$ | -19.99 | -0.87 | -19.99 | -19.82 |
| $\hat{p}$ | 1.08 | 1.08 | 1.08 | 1.08 |
| $\hat{A}_p \times 10^{-3}$ | 7.77 | 5.22 | 7.77 | 5.22 |
| $\hat{\varphi}_p$ | -23.13 | -16.59 | -23.13 | -7.41 |

Table 5.17: NASA Rotor 67: exact and scaled mean values of state variables, amplitudes and phase angles.

The same number of modes used for the on-reference cases was used for the off-reference cases to produce a solution. A FOM solution was generated for assessing the accuracy of the ROM solution. This FOM solution was not used to generate basis functions. Figure 5.21 shows the time-averaged relative error between the off-reference ROM solution for pressure and the FOM solution for pressure for case 4. Pressure had the largest error, approximately 3%, that occurred near the tip of the airfoil where a shock developed. Figure 5.22 shows the time-averaged relative error between the off-reference ROM and FOM state variables for case 5. The largest errors occurred at the outlet, as opposed to the shock location, and their values were smaller than 0.5%.

Table 5.18 gives the maximum spatial and temporal relative error between the off-reference

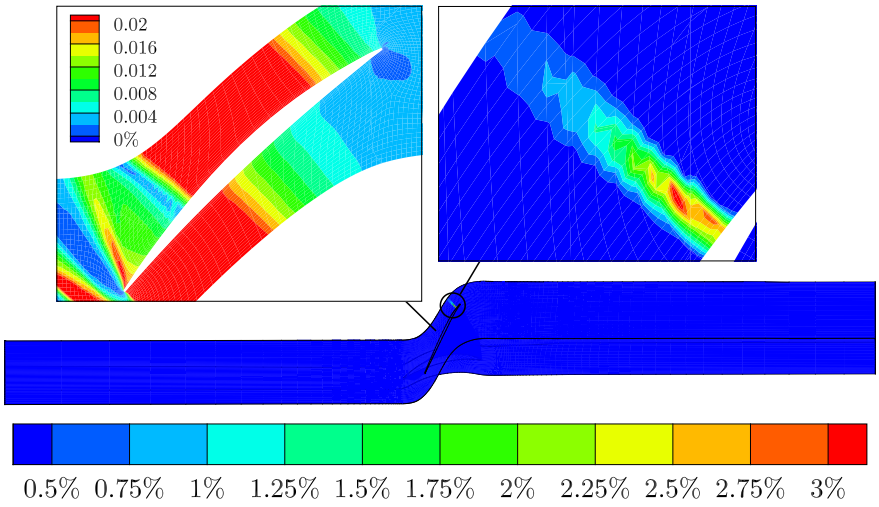
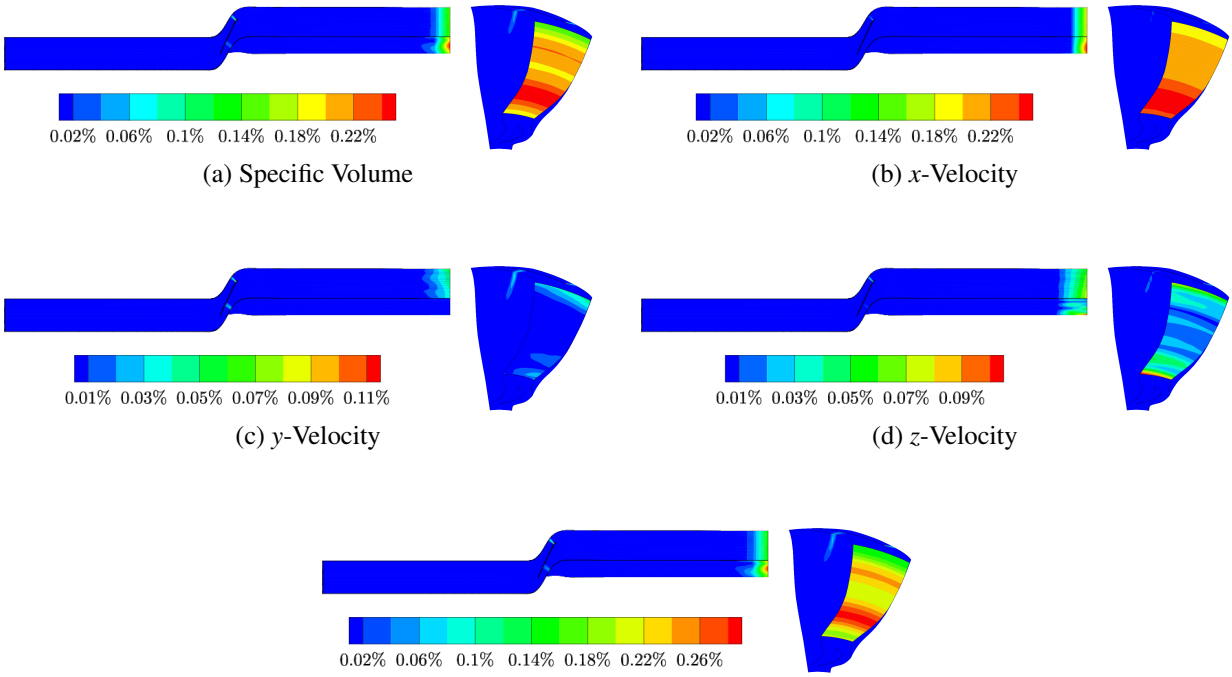



Figure 5.21: NASA Rotor 67: time-averaged relative error between the off-reference ROM and FOM pressures for an outlet pressure oscillating with angular velocity $\tilde{\omega} = 38,059$ rad/sec and amplitude $\tilde{A} = 0.075$. A close-up of the error around the blade is shown at blade midspan while a close-up of the error around the shock is shown at around 95% blade span location measured from the hub. Note that the legend for the midspan contour plots is different from the other two plots.

ROM and FOM state variables for cases 4 and 5. The small error implies that the zeta-variable ROM accurately predicted the off-reference solutions even when using the approximate boundary conditions. The zeta-variable ROM captured well the shock wave that formed on the suction side of the blade.

| | $\max_{x,t}(\epsilon)$ (%) | | | | |
|---|---|---|---|---|---|
| Case | $\zeta$ | $u$ | $v$ | $w$ | $p$ |
| 4 | 1.85 | 0.97 | 1.26 | 0.82 | 3.46 |
| 5 | 0.57 | 0.41 | 0.12 | 0.17 | 0.68 |

Table 5.18: NASA Rotor 67: maximum spatial and temporal relative error between the off-reference ROM and FOM state variables.




(a) Specific Volume (b) $x$-Velocity

(c) $y$-Velocity (d) $z$-Velocity

(e) Pressure

Figure 5.22: NASA Rotor 67: time-averaged relative error between the off-reference ROM and FOM state variables, for an outlet pressure oscillating with angular velocity $\tilde{\omega} = 37,765$ rad/sec and amplitude $\tilde{A} = 0.05$.

### 5.5 10th Standard Configuration

This section presents on- and off-reference results for the blade deformation of the 10th standard configuration, which is a modified NACA 0006 airfoil at a stagger angle of $45°$ [77, 78]. The flow enters the inlet with an angle of attack of $55°$ at Mach 0.7. The stagnation pressure is 140.55 kPa, the stagnation temperature is 316.39 K, and the static outlet pressure is 122.05 kPa. Figure 5.23 shows its computational grid with $N = 33,068$ nodes.

Flow unsteadiness is introduced by the blade plunging with an amplitude

$$y_{\text{forced}}(t) = \bar{y}\left(1 + \tilde{A}\sin(2\pi f t)\right) \tag{5.17}$$

where $\bar{y}$ is the $y$-location of the surface of the blade at its undeformed state and $f$ is the frequency.

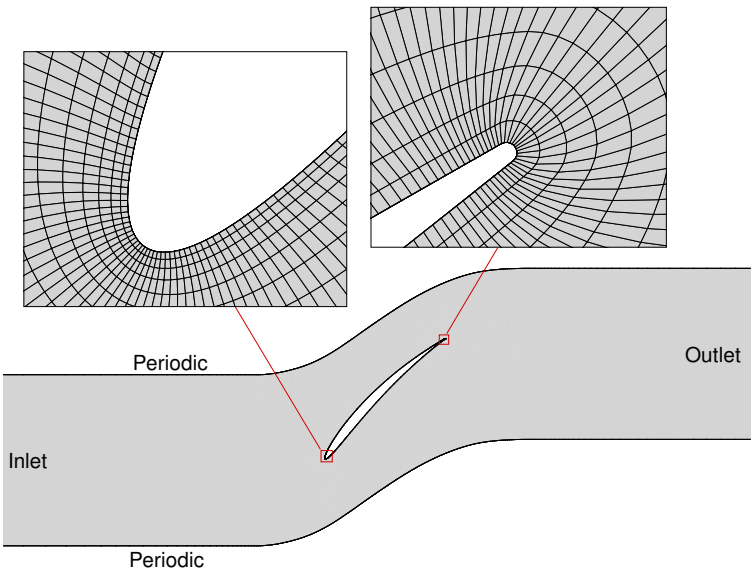



Figure 5.23: 10th standard configuration computational mesh

Radial basis functions were used to deform the mesh [79]. The amplitude was varied across three cases as summarized in Table 5.19. Cases 1 and 2 are on-reference cases and case 3 is an off-reference case. The FOM produced snapshots for each case for 5 periods with 100 snapshots per period.

| Case | Plunge Amplitude [-] | Frequency [Hz] | Type |
|---|---|---|---|
| 1 | 0.02 | 10 | on-reference |
| 2 | 0.04 | 10 | on-reference |
| 3 | 0.03 | 10 | off-reference |

Table 5.19: 10th standard configuration on- and off-reference cases

The penalty method was imposed onto the ROM to enforce the boundary deformation of the blade. To impose the boundary deformation, a point on the blade was selected to create the prescribed boundary condition, $\mathbf{F}(t)$. This point was located at $x/c \approx 0.18$ on the lower side of the blade. For this problem, the prescribed boundary condition was found by finding the Fast

Fourier Transform (FFT) approximation of the FOM solution at the selected boundary node. For off-reference conditions, the coefficients of the FFT approximation were linearly interpolated to the new flow conditions.

The error between the ROM and the FOM was calculated using (5.5) and (5.6). Figure 5.24 shows the the maximum relative error of $x$-velocity for all time and space for case 1. The maximum error occurs at the trailing edge of the blade. This was the same location for all on-reference cases.

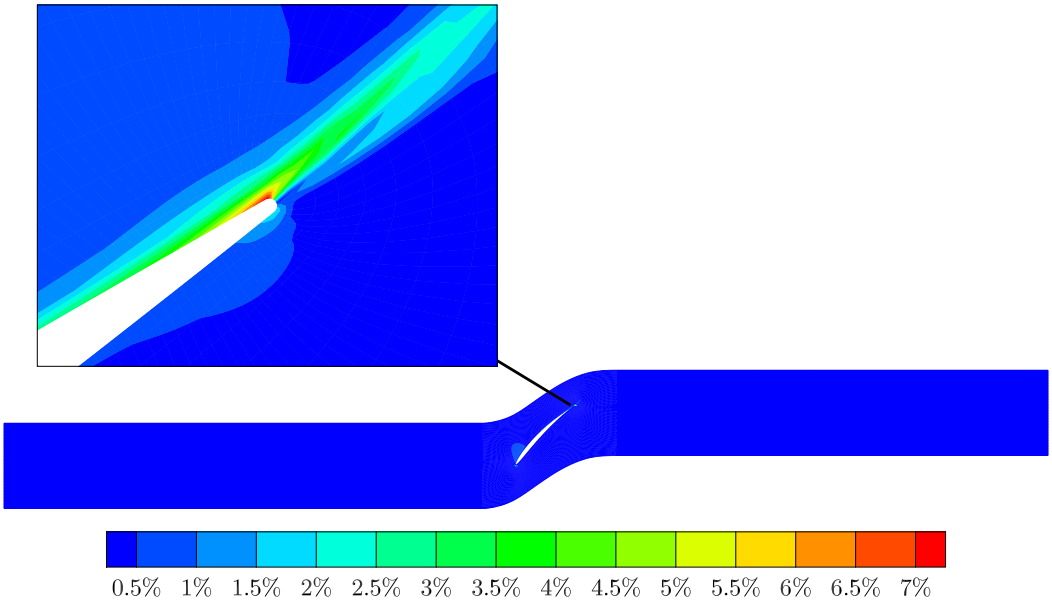



Figure 5.24: 10th standard configuration: location of maximum error for $x$-velocity for case 1.

The maximum error occurs around the trailing edge of the blade where the grid is coarse relative to the changes in the flow dynamics. Increasing the grid quality in that area may reduce the errors.

Table 5.20 summarizes the spatial and time-averaged relative error, the spatial-averaged maximum relative error in time, and the maximum relative error for all time and space for all variables. The basis functions for case 3 were developed by interpolating the basis functions of cases 1 and 2 using Grassmann interpolation. It is important to note that the FOM solution for case 3 was generated for comparison purposes only, and was not used to produce the basis functions the ROM

used. The largest error is around $18\%$, which occurred near the trailing edge of the blade. Despite this large error, the ROM was able to accurately reproduce (for cases 1 and 2) and predict (for case 3) the FOM solution at the given flow conditions with the average maximum error in the domain being less than $1\%$.

| Case | Variable | *x,t*-Avg Error (%) | *x*-Avg Max Error in Time (%) | Max Error (%) |
|---|---|---|---|---|
| 1 | $\zeta$ | 0.15 | 0.32 | 5.99 |
| | $u$ | 0.11 | 0.26 | 7.35 |
| | $v$ | 0.20 | 0.42 | 3.40 |
| | $p$ | 0.21 | 0.44 | 2.51 |
| 2 | $\zeta$ | 0.31 | 0.68 | 5.46 |
| | $u$ | 0.21 | 0.48 | 9.47 |
| | $v$ | 0.38 | 0.79 | 4.56 |
| | $p$ | 0.42 | 0.88 | 4.90 |
| 3 | $\zeta$ | 0.23 | 0.49 | 5.00 |
| | $u$ | 0.20 | 0.52 | 17.68 |
| | $v$ | 0.31 | 0.75 | 11.67 |
| | $p$ | 0.34 | 0.87 | 8.94 |

Table 5.20: Relative error of the ROM to the FOM for the 10th standard configuration.

### 5.6 11th Standard Configuration

Results here are presented for the 11th standard configuration, which is a turbine cascade with a $180^\circ$ inter blade phase angle [78, 80]. Figure 5.25 shows a detail of the computational grid used with $N$ = 78,260. UNS3D POD generated snapshots for laminar flow with an inlet flow angle of $15.2^\circ$ and a back pressure of 78.8 kPa. The inlet stagnation pressure and temperature are 108.31 kPa and 293.8 K, respectively. A total of five-hundred snapshots were taken with 50 points per period to develop the auto-correlation matrix of Eq. (2.5).

To impose unsteadiness in the flow, the blades plunged normal to the chord with a deformation given by

$$x(t_i) = x(t_{i-1}) + \tilde{A}\cos(2\pi f t)\sin(\hat{\gamma})$$

$$y(t_i) = y(t_{i-1}) + \tilde{A}\cos(2\pi f t)\cos(\hat{\gamma})$$

where $\hat{\gamma} = -40.85°$ is the stagger angle of the blades. Table 5.21 summarizes the cases for this configuration. The amplitude and reduced frequency were varied between cases to change the deformation. The reduced frequency, $k$, was calculated based on the semi chord and the exit velocity. Cases 1, 2, and 4 are on-reference, that is, cases for which the basis functions are directly derived from a FOM solution by solving Eq. (2.6). Cases 3 and 5 are off-reference, that is, cases, for which the basis functions are not directly derived from a FOM solution. Interpolation on the tangent space to a Grassmann manifold was used to find the basis functions for cases 3 and 5 [45]. The basis functions of cases 1 and 2 were interpolated to find the basis functions for case 3, and the basis functions of cases 1 and 4 were interpolated to find the basis functions for case 5.

To impose boundary conditions into the zeta-variable ROM, the penalty method was used [39, 40]. The prescribed boundary conditions $\mathbf{F}(t) = (\mathbf{F}^{\zeta}, \mathbf{F}^{u}, \mathbf{F}^{v}, \mathbf{F}^{p})^{T}$ were found by fitting an eight-

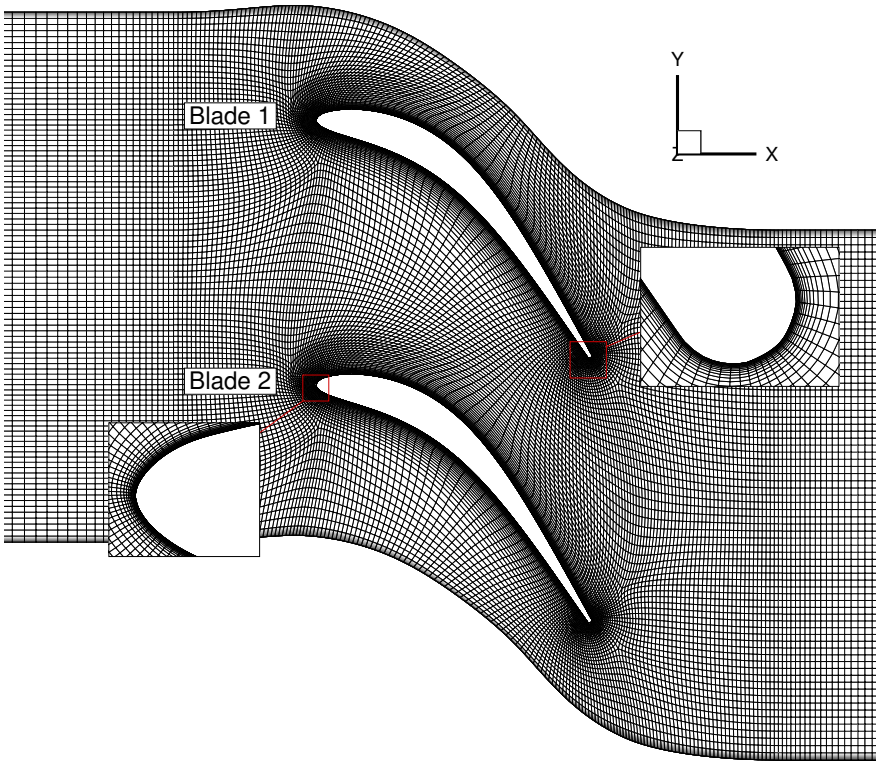



Figure 5.25: 11th standard configuration computational grid.

| Case | Type | Amplitude$\times 10^{-4}$ | Frequency [Hz] | $k$ |
|---|---|---|---|---|
| 1 | On-reference | 3.7811 | 209 | 0.45 |
| 2 | On-reference | 4.6213 | 209 | 0.45 |
| 3 | Off-reference | 4.2012 | 209 | 0.45 |
| 4 | On-reference | 3.7811 | 327 | 0.70 |
| 5 | Off-reference | 3.7811 | 281 | 0.60 |

Table 5.21: 11th Standard Configuration Cases

term Fourier Series model to time-series data of the FOM solution. The time-series data were collected from a point along both blades corresponding to $x/c \approx 0.26$ on the suction side of the blade and then averaged together. This means that $N_{\text{BC}} = 1$ for this case. The penalty parameter was solved for at each time step using Eq. (4.7). For off-reference conditions, the Fourier coefficients were linearly interpolated based off of either the amplitude or frequency change.

The zeta-variable ROM ran for 500 iterations for each case and each case was compared to its respective FOM solution. For cases 3 and 5, a reference FOM solution was generated for comparison purposes. It is important to note that this reference FOM solution was not used to generate basis functions for the ROM.

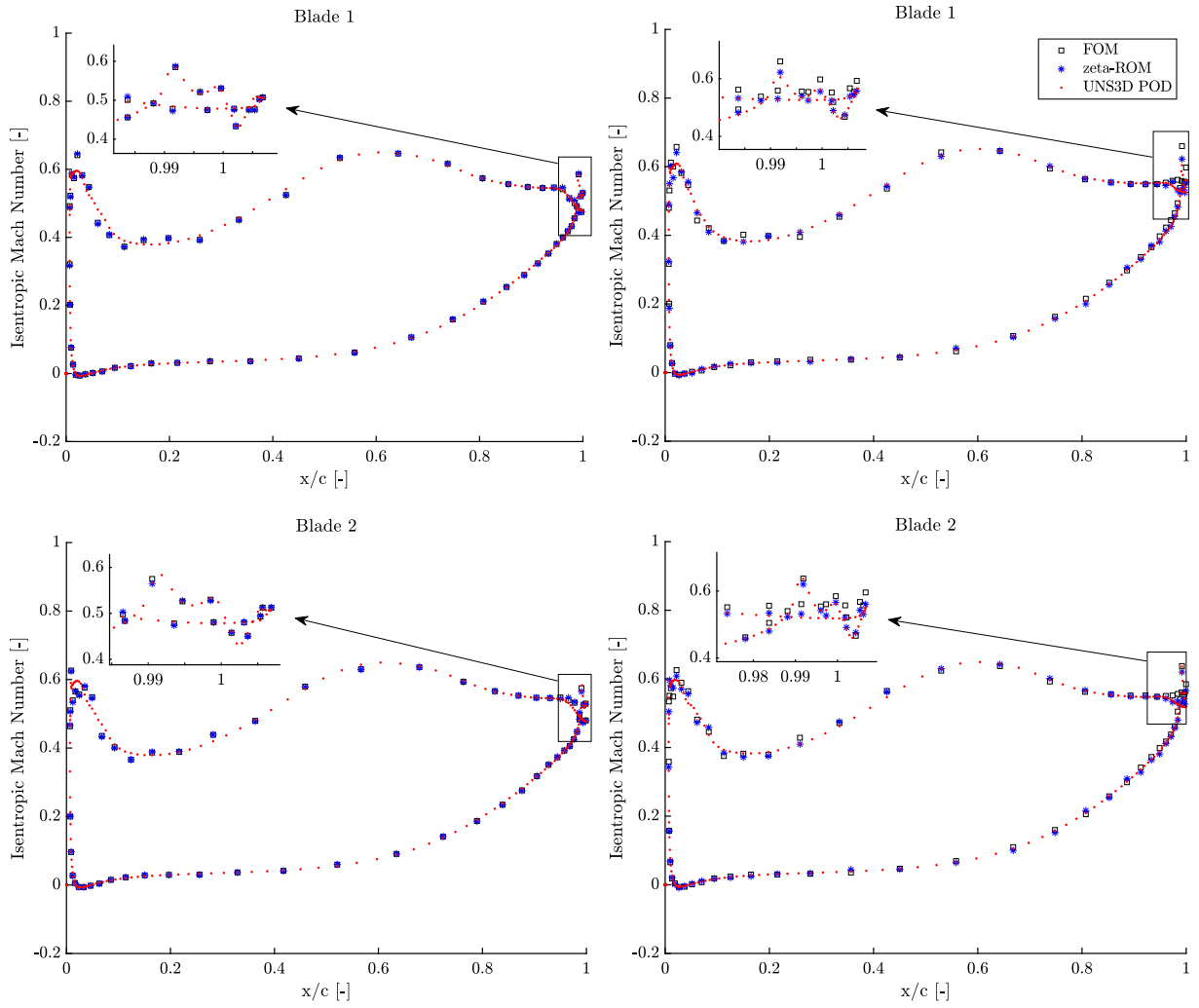
To compare the ROM solution to the FOM solution, (5.5) and (5.6) were used. Table 5.22 gives the spatial and time-averaged relative error, the spatial-average of the maximum relative error in time, and the maximum relative error for all time and space of the ROM to the FOM for all cases. For cases 1, 2, and 4, which are the on-reference cases, the zeta-variable ROM does fairly well with a maximum relative error of less than 5%. For cases 3 and 5, there are large maximum relative errors. These errors occur near the trailing edge of the blades. It is possible that some of the error is due to the interpolation of the Fourier series coefficients and some to the interpolation of the basis functions. Overall, as can be seen by the averages of the relative error, the relative error remains low throughout most of the domain.

The zeta-variable POD ROM was also compared to UNS3D POD. Figure 5.26 shows a comparison of the isentropic Mach number between the zeta-variable ROM and UNS3D POD as compared

| Case | Variable | $\text{avg}_{x,t}(\epsilon)$ (%) | $\text{avg}_x\left(\max_t\left(\epsilon\right)\right)$ (%) | $\max_{x,t}(\epsilon)$ (%) |
|---|---|---|---|---|
| 1 | $\zeta$ | 0.03 | 0.06 | 1.23 |
| | $u$ | 0.04 | 0.10 | 3.41 |
| | $v$ | 0.02 | 0.07 | 1.96 |
| | $p$ | 0.03 | 0.06 | 0.76 |
| 2 | $\zeta$ | 0.04 | 0.07 | 1.34 |
| | $u$ | 0.06 | 0.12 | 2.95 |
| | $v$ | 0.02 | 0.07 | 2.30 |
| | $p$ | 0.03 | 0.06 | 0.76 |
| 3 | $\zeta$ | 0.27 | 0.60 | 12.20 |
| | $u$ | 0.02 | 0.07 | 3.57 |
| | $v$ | 0.02 | 0.07 | 2.15 |
| | $p$ | 0.03 | 0.06 | 0.64 |
| 4 | $\zeta$ | 0.04 | 0.09 | 1.65 |
| | $u$ | 0.06 | 0.14 | 4.47 |
| | $v$ | 0.03 | 0.10 | 3.10 |
| | $p$ | 0.03 | 0.08 | 1.29 |
| 5 | $\zeta$ | 0.37 | 0.71 | 10.81 |
| | $u$ | 0.43 | 0.85 | 21.72 |
| | $v$ | 0.58 | 1.04 | 24.45 |
| | $p$ | 0.27 | 0.65 | 5.85 |

Table 5.22: Relative error of the ROM to the FOM for the 11th standard configuration plunging blades with $180^\circ$ inter blade phase angle.

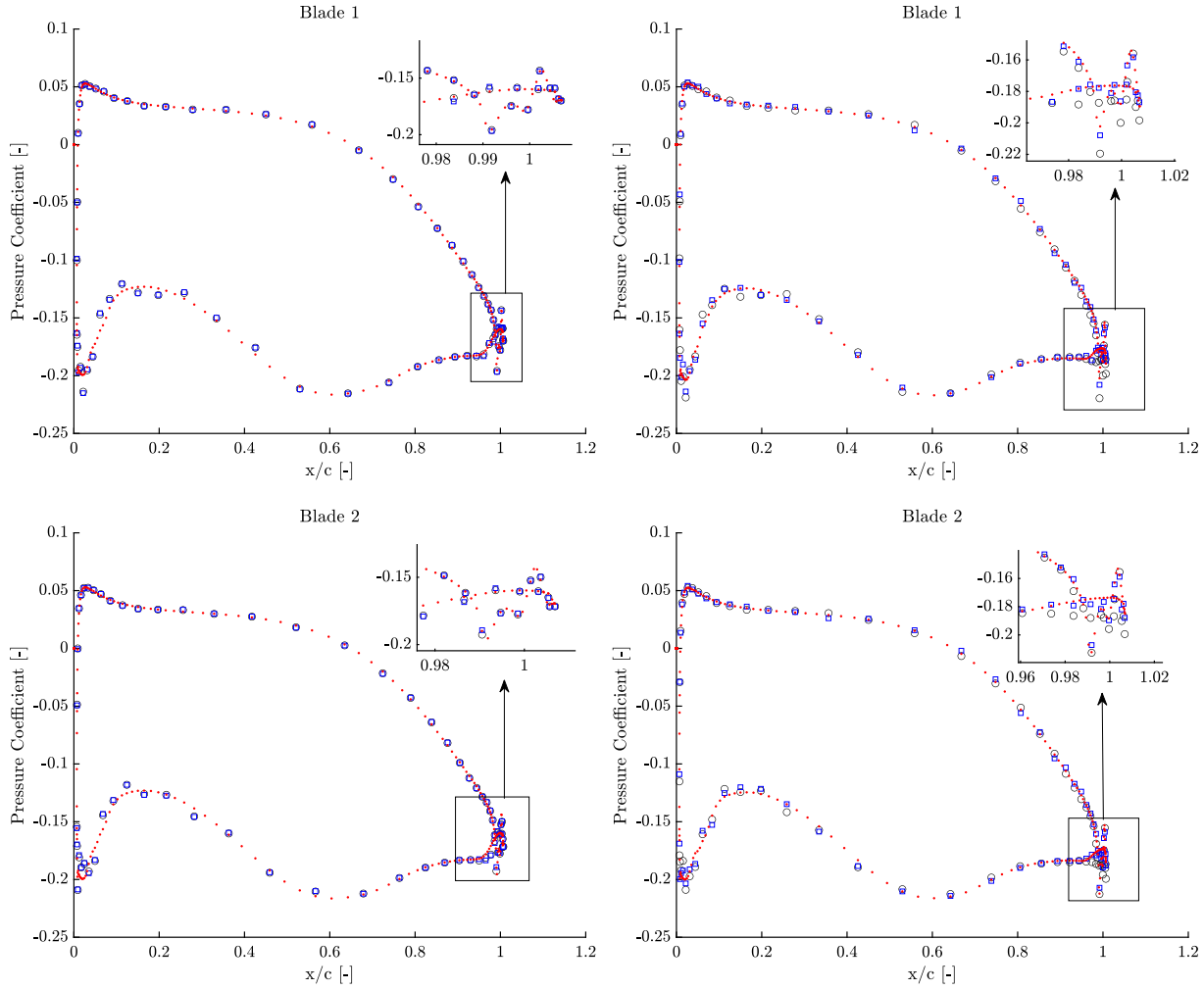
to the FOM for cases 3 and 5. A closeup of the trailing edge is provided in each figure. Along the interior of the airfoil, both the zeta-variable ROM and UNS3D POD solutions match well with the FOM solution. Towards the leading edge on the suction side of the blade, UNS3D POD under-predicts the isentropic Mach number. At the trailing edge of the blade, the zeta-variable ROM and UNS3D POD solutions match well with each other, but stray from the FOM solution. The zeta-variable ROM appears to be more accurate than UNS3D POD for this case. Figure 5.27 shows a comparison of the coefficient of pressure between the zeta-variable ROM and UNS3D POD as compared to the FOM for cases 3 and 5. Again, UNS3D POD under-predicts at the leading edge of the suction side of the blade whereas the zeta-variable ROM matches well with the FOM. Both ROMs match well with the FOM solution along the interior of the blade. At the trailing edge there




(c) Case 3 (d) Case 5

Figure 5.26: Comparison of the isentropic Mach number of the zeta-variable ROM to UNS3D POD to the FOM.

is a difference in the results between cases 3 and 5. For case 3, the zeta-variable ROM and UNS3D POD match well with the FOM solution. For case 5, both ROM solutions match well with each other, but do not match well with the FOM solution. The results of these comparisons show that both the zeta-variable ROM and UNS3D POD experience larger errors near the leading and trailing edges of the blades. As mentioned before, the zeta-variable ROM incurred the largest of its errors at the trailing edge of the blade, which accounted for the large maximum errors seen in Table 5.22. This may imply that a majority of the error lies within the interpolation of the basis functions rather




(c) Case 3 (d) Case 5

Figure 5.27: Comparison of the coefficient of pressure of the zeta-variable ROM to UNS3D POD to the FOM. The legend follows that of Figure 5.26.

than the methods of the ROMs themselves. The zeta-variable ROM may have performed better than UNS3D POD at the leading edge due to its use of the penalty method to impose boundary condition constraints.

# 6. STABILITY ANALYSIS OF THE ZETA-ROM

This section provides a stability analysis of the zeta-ROM using Lyapunov's method from sec. 4.1.1. Results are shown for only the 2D channel. There were shortcomings for using this method to investigate stability, which will be discussed at the end of this section.

The ODE system of the ROM is a nonlinear system of equations. When using the penalty method, the system becomes non-autonomous. This makes it difficult to assess how well the penalty method is stabilizing the ROM. Lyapunov's method can be used to assess the stability of nonlinear, non-autonomous systems as long as specific criteria are met [69].

A simple Lyapunov's function was selected to assess the stability of the zeta-ROM. Here, the function is described by the time coefficients, **a**.

$$V(\mathbf{a}) = \sum_{i=1}^{n^{Z_k}} \left(a_i^{Z_k}\right)^2, \ k \in [1, D+2] \tag{6.1}$$

This function resembles the kinetic energy of the system, which makes it ideal for assessing whether energy in the system is decreasing or increasing. For completeness, the time-derivative of the Lyapunov's function is

$$\dot{V}(\mathbf{a}) = 2\sum_{i=1}^{n^{Z_k}} a_i^{Z_k}\dot{a}_i^{Z_k}, \ k \in [1, D+2]. \tag{6.2}$$

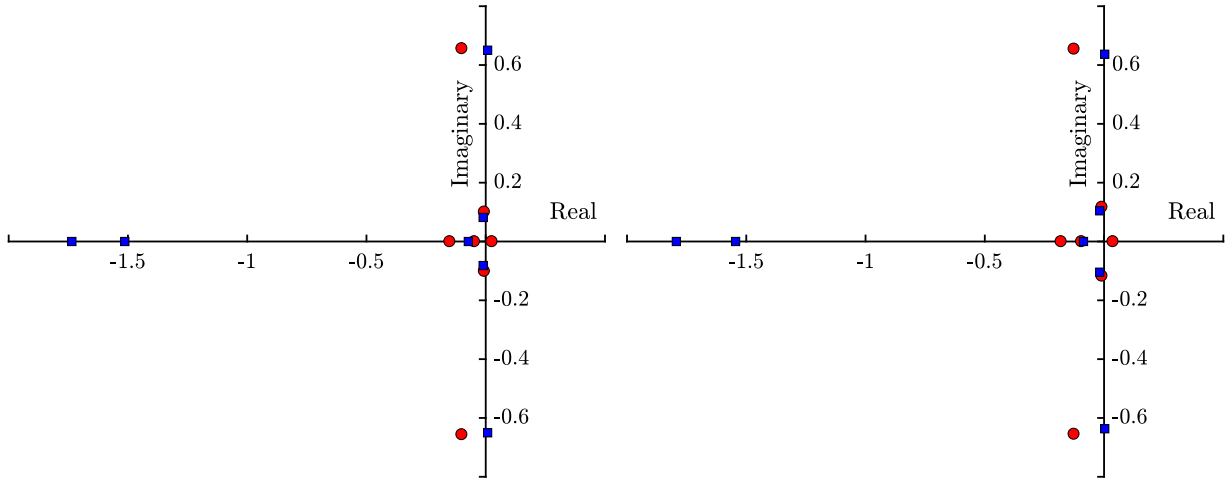
Table 6.1 shows the effect of the penalty method and a modified number of modes on the stability of the zeta-ROM for the 2D channel. The cases reflect the cases in Table 5.4. Case 2 has no results, because the critical points were unable to be found. Based off of the rate of the Lyapunov's function, the implementation of the penalty method along with a modified number of modes appears to stabilize the zeta-ROM.

Figure 6.1 shows the effect of the penalty method along with a modified number of modes on the eigenvalues. For both cases, a positive real part of an eigenvalue appears, which indicates instability. When the penalty method along with a modified number of modes are applied, the

| Case | $\dot{V}(\mathbf{a})$ | |
|---|---|---|
| | No Method | Penalty Method with Modified Number of Modes |
| 1 | 0.31 | -1.63 |
| 3 | 0.06 | -8.22 |

Table 6.1: Effect of the penalty method and a modified number of modes on $\dot{V}(\mathbf{a})$.

eigenvalues shift to a complex conjugate pair. It is important to note that this case ran successfully with results reported in Sec. 5.2 once the penalty method with a modified number of modes was applied. Therefore, there appears to be an issue with using Lyapunov's first method to approximate stability for this case. This is due to the system being highly nonlinear and non-autonomous.




(a) Case 1 (b) Case 3

Figure 6.1: Effect of the penalty method and a modified number of modes on the eigenvalues of the zeta-ROM ODE system. The red markers denote the zeta-ROM without stabilizing methods imposed. The blue markers denote the zeta-ROM with stabilizing methods imposed.

Stability analysis using Lyapunov's method failed for all other cases, because, similar to what happened with case 2, the critical points were unable to be found. Without the critical points, the stability analysis is unable to be conducted. While it is difficult to determine *a priori* the stability of the zeta-ROM ODE system, analysis using the Lyapunov's function did indicate some measure of

success. Therefore, more investigation into this method or possibly others designed for nonlinear, non-autonomous systems should be performed.

# 7. COMPUTATIONAL TIME RESULTS[1]

This section presents a comparison between the runtimes of the zeta-variable ROM and the FOM for all cases. The reduced-order model was ran on a 2009 Mac Pro with two 2.26 GHz Quad-Core Intel Xeon processors with 24 GB of RAM. The full-order model was ran on both the 2009 Mac Pro and on Texas A&M's Terra supercomputing cluster and used multiple processors. The CPU run time is given to account for the use of parallel processing for the FOM.

Table 7.1 shows the offline costs of running the FOM and generating the basis functions. Here, (P) and (D) for the axisymmetric nozzle refer to the cases with a pressure oscillation and the cases with a deforming boundary, respectively. The CPU time of the snapshot generation phase in the FOM, the CPU time it took to run the FOM, and the CPU time to calculate the POD basis functions are given. It is worth it to note that the total time of the FOM does not include the time to calculate the POD basis functions. The snapshot generation phase takes up a majority of the run time of the FOM as the transients in the solution quickly disappear. For Rotor 67 and the 3D Nozzle (D), the snapshot generation phase takes up a smaller portion due to long transients in the solution. Generating the POD basis functions from the snapshots takes up a minuscule part, less than 1%, of the runtime of the offline phase when compared to the run time of the FOM.

Table 7.2 shows a comparison between the CPU runtime of the zeta-variable ROM and the FOM for all cases. The FOM solutions were generated using the UNS3D flow solver [59]. The FOM solution of the two-dimensional channel was obtained running UNS3D as an implicit solver while all other case solutions were obtained by running UNS3D as an explicit solver. The FOM and the ROM were run such that they both predicted the flow over the same time interval. The FOM used six processors for the 2D channel and 3D nozzle cases and used 56 processors for Rotor 67 and the 10th standard configuration cases. From the data, the ROM is consistently more than four orders of magnitude faster than the FOM. The computational time needed to generate the POD

[1] A portion of this section is reprinted from "An Efficient Proper Orthogonal Decomposition based Reduced-Order Model for Compressible Flows" by E.H. Krath, F.L. Carpenter, P.G.A. Cizmas, and D.A. Johnston, 2020. *Journal of Computational Physics*. Copyright 2020 by Elsevier Inc. https://doi.org/10.1016/j.jcp.2020.109959

| Case | 2D Channel | 3D Nozzle (P) | 3D Nozzle (D) | Rotor 67 |
|---|---|---|---|---|
| $N$ | 8514 | 22,673 | 22,673 | 299,844 |
| Snapshots per Period | 300 | 666 | 100 | 50 |
| Total Snapshots | 900 | 2000 | 300 | 250 |
| FOM Snapshots [s] | 15,130.75 | 506,635.00 | 61,049.33 | 388,891 |
| FOM Total [s] | 15,664.75 | 508,018.00 | 164,872.68 | 4,860,247 |
| POD Basis Functions [s] | 121.33 | 1673.00 | 50.04 | 479.43 |
| POD Basis/FOM Total | 0.84% | 0.33% | 0.02% | 0.01% |
| | 10 STCF | 11th STCF | | |
| $N$ | 33,068 | 78,260 | | |
| Snapshots per Period | 100 | 100 | | |
| Total Snapshots | 500 | 500 | | |
| FOM Snapshots [s] | 473,059 | 2,260,688.25 | | |
| FOM Total [s] | 530,558 | 10,883,754.00 | | |
| POD Basis Functions [s] | 172.63 | 397.365 | | |
| POD Basis/FOM Total | 0.03% | 0.004% | | |

Table 7.1: Breakdown of the CPU runtime of the FOM for three on-reference cases: (1) 2D channel with 7 modes, (2) 3D nozzle (P) with 9 modes, and (3), 3D nozzle (D) with 5 modes.

basis functions was not included in this comparison.

| Case | $N$ | $M$ | FOM [s] | ROM [s] | Ratio |
|---|---|---|---|---|---|
| 2D Channel | 8514 | 900 | 15,131 | 0.52 | 29,098 |
| 3D Nozzle (P) | 22,673 | 2000 | 506,635 | 5.38 | 94,170 |
| 3D Nozzle (D) | 22,673 | 300 | 61,049 | 4.19 | 14,564 |
| Rotor 67 | 299,844 | 250 | 4,860,247 | 51.30 | 94,749 |
| 10 STCF | 33,068 | 500 | 530,558 | 12.46 | 42,580 |
| 11 STCF | 78,260 | 500 | 2,260,688 | 24.35 | 92,841 |

Table 7.2: CPU runtime comparison between the ROM and the FOM for all cases: (1) 2D channel with 7 modes, (2) 3D nozzle (P) with 9 modes, (3) 3D nozzle (D) with 5 modes, (4) Rotor 67 with 5 modes, and (5) the 10th standard configuration with 5 modes.

Table 7.3 breaks down the run time of the ROM into two segments: the setup, which is pre-

calculating the inner products, and the time it takes to solve the ODEs for 5 periods. A theoretical upper limit of speedup can be created by finding the ratio of the run time of the FOM to the time it takes to solve the ODEs.

| | | ROM | | | Upper | % Upper |
|---|---|---|---|---|---|---|
| Case | No. Periods | Setup [s] | Solve ODE [s] | Total [s] | Limit | Limit |
| 2D Channel | 3 | 0.22 | 0.08 | 0.30 | 195,787 | 15% |
| 3D Nozzle (P) | ≈3 | 2.34 | 3.04 | 5.38 | 166,394 | 57% |
| 3D Nozzle (D) | 3 | 1.93 | 2.26 | 4.19 | 26,997 | 54% |
| Rotor 67 | 5 | 19.45 | 31.84 | 51.30 | 152,623 | 62% |
| 10 STCF | 5 | 5.97 | 6.49 | 12.46 | 81,773 | 52% |
| 11 STCF | 5 | 11.93 | 12.42 | 24.35 | 182,020 | 51% |

Table 7.3: Breakdown of CPU run time of ROM and comparison to theoretical upper limit of speedup.

The reduced-order model currently runs at an average of 49% of this upper limit. The 2D channel case has a large setup time compared to the time it takes to solve the ODE, which leads to a lower percent upper limit achieved. Improvements can be made by reducing the time of the setup phase.

Additionally, the upper limit of the speedup can be improved by reducing the time it takes to solve the ODEs. Part of this was already accomplished by rewriting the governing equations using specific volume instead of density. Changing the form of the governing equations was used to prevent any slowdown while solving the ODEs. A slowdown would have occurred if the coefficients of the ODEs had to be recalculated at each time step or if a separate system of equations had to be solved to find the time-derivative of the time coefficients. It is also good to note that the run time of the setup phase is independent of the length of time that the ROM solves for. For increasing lengths of time or number of periods, the speedup of the ROM will be entirely dependent on the time it takes to solve the ODE system.

To better assess the efficiency of the zeta-variable ROM, it is better to compare its performance

to other ROMs that do not use zeta-variables. The zeta-ROM was compared to the UNS3D POD ROM for the 2D channel and the 11th standard configuration.

Table 7.4 shows a comparison of the CPU runtime of the zeta-variable ROM to the conservative-variable ROM for the 2D channel. Because UNS3D POD is based off of the discretized scheme of the governing equations, it depends on certain factors related to the FOM such as the CFL number and the number of sub-iterations used in dual time stepping $M_{\text{sub}}$. These two factors were varied to show how the run time of the conservative-variable ROM changes as compared to the zeta-variable ROM. The results of the runtime are also shown for the conservative-variable ROM using an implicit and explicit solver. The zeta-variable ROM used the same solver from the ODEPACK suite as used in all other cases. The results show a marked improvement in runtime from the conservative-variable ROM to the zeta-variable ROM, with the zeta-variable ROM being more than 10,000 times faster than the conservative-variable ROM. This speed up is due in large part to removing the occurrence of density in the denominator, which allowed for the pre computation of the coefficients of the ODEs that make up the ROM.

| Solver | $m$ | CFL | $M_{\text{sub}}$ | Conservative [s] | Zeta [s] | Ratio |
|---|---|---|---|---|---|---|
| I | 2 | 20 | 15 | 7423 | 0.40 | 12,200 |
| E | 2 | 3 | 150 | 29,851 | 0.39 | 76,541 |
| | 2 | 5 | 150 | 29,644 | 0.39 | 76,010 |
| | 2 | 5 | 140 | 27,978 | 0.39 | 71,738 |
| | 2 | 5 | 130 | 26,001 | 0.39 | 66,669 |
| | 2 | 15 | 150 | 29,609 | 0.39 | 73,356 |

Table 7.4: CPU runtime comparison of the zeta-variable ROM to a conservative-variable ROM for different solvers, CFL numbers, and sub-iterations for a 2D channel case with $N = 4257$ and $m = 6$.

Table 7.5 shows the shows a comparison of the average CPU run time for the 11th standard configuration case. Without post-processing, the zeta-variable ROM will only output the time coefficients at each time step. Whereas, with post-processing, the zeta-variable ROM will also

output the relative errors, which requires the reconstruction of each state variable at each time step. Here, UNS3D POD and the FOM were ran on Texas A&M University's supercomputing cluster Ada, which is an Intel x86-64 Linux cluster. The ratios provided in Table 7.5 compare the CPU runtime of the FOM or UNS3D POD to the zeta-variable ROM. The zeta-variable ROM ran more than five orders of magnitude faster than the FOM and more than three orders of magnitude faster than UNS3D POD. When post-processing time is included in the zeta-variable ROM, the size of the speedup drops by about an order of magnitude.

| Post-Process | zeta-ROM [s] | FOM $\times 10^6$ [s] | Ratio | UNS3D POD$\times 10^4$ [s] | Ratio |
|---|---|---|---|---|---|
| No | 24.35 | 2.26 | 92,841.41 | 7.51 | 14,625.24 |
| Yes | 189.33 | 2.26 | 11,940.47 | 7.51 | 1880.85 |

Table 7.5: Computational runtime with and without additional post-processing of the zeta-variable POD ROM (zeta-ROM) versus UNS3D POD and the FOM for the 11th standard configuration plunging blades with $180^\circ$ inter blade phase angle.

### 7.1 Cost-benefit of POD-based ROM

It is important to quantify when a reduced-order model is beneficial to use as compared to the full-order model. Using a ROM to reconstruct an already generated full-order model solution, at on-reference conditions, does not add any new information. A ROM truly becomes beneficial when it is used to predict new flow solutions at off-reference conditions. To use the ROM at off-reference conditions, the basis functions needed to be interpolated. The methods used to interpolate the basis functions were given in Sec. 4.2.

For all cases shown, the ROM needed two FOM snapshot sets to solve at off-reference conditions. The two FOM snapshot sets bounded a parameter of interest, for example, the amplitude or angular velocity of an imposed oscillatory motion. With these two FOM snapshot sets, the ROM could predict any number of solutions in between the bounds. Let's say that three solution sets are needed for a change in a parameter of interest $\beta$ on the 2D channel case in Table 7.2. If only the FOM

is used, then the total time is $3 \times 15,131$ sec $= 45,393$ sec. If the ROM is used, then two FOM solutions are needed at first, $2 \times 15,131$ sec $= 30,262$ sec, plus one ROM run, $0.52$ sec, which is a total of $30,262.52$ sec or $1.5\times$ faster than only using the FOM. Therefore, for this case the ROM becomes beneficial to use if three or more flow solutions are needed.

Let's say instead that there are multiple parameters of interest, $\boldsymbol{\beta} = \{\beta_i\}_{i=1}^{n_\beta}$, where $n_\beta$ is the number of parameters of interest. Assuming a linear model is used to interpolate the basis functions to a specified design point, $\boldsymbol{\beta}^*$, then the number of FOM evaluations needed to create the linear model is $n_\beta + 1$. Therefore, the ROM becomes beneficial to use for multivariate cases if $n_\beta + 2$ solutions are needed.

# 8. CONCLUSIONS

An efficient Proper Orthogonal Decomposition based reduced-order model for compressible flows was developed and verified and validated for several cases of varying complexity. This reduced-order model, or zeta-ROM, was made efficient by substituting in specific volume for density in the governing equations, which allowed for the pre computation of the coefficients and reduced the nonlinearity of the ROM ODE system. The zeta-ROM was able to accurately reconstruct and predict flow solutions with aeroelastic deformations. This included a radial deformation of an axisymmetric nozzle and a plunging turbine blade for the 10th and 11th standard configurations. The accuracy of the zeta-ROM was further enhanced by using either a dynamic average or dynamic basis functions. Furthermore, the zeta-ROM was able to accurately reconstruct and predict flow solutions for cases with a pressure oscillation applied at the outlet.

The instabilities of the model reduction procedure were apparent in the zeta-ROM. Three methods were developed and successfully used to stabilize the zeta-ROM. These were the penalty method, artificial dissipation, and a new method that modifies the number of modes in the ROM ODE system.

The penalty method worked very well to enforce boundary conditions into the ROM. Surprisingly, it was also an effective tool when using the zeta-ROM to predict off-reference solutions. Using the penalty method for off-reference conditions constrained the solution to match the target design parameters, which increased the accuracy of the zeta-ROM.

Artificial dissipation was helpful to stabilize the zeta-ROM, but the artificial dissipation parameters can become tricky to tune. For this reason, a method to calculate the artificial dissipation parameters such that the energy in the system is decreasing was developed. This was made especially helpful for off-reference conditions when the artificial dissipation parameters are not able to be tuned against a full-order model solution.

The method that modifies the number of modes, or referred to as “modified number of modes”, in the ROM ODE system was effective at removing destabilizing modes from the ODE system. This

method came about by observation. Spurious modes can sometimes be captured using the POD method. For some cases, these spurious modes can be destabilizing. This method acted as a filter to remove these modes, and was successful in further stabilizing the zeta-ROM.

A stability analysis was performed of the zeta-ROM. This analysis was performed with the hopes of determining *a priori* the effects of the stabilizing methods on the ROM before solving the ODE system. Lyapunov's first and second methods were used with varying degrees of success. Because the ODEs of the zeta-ROM are nonlinear and non-autonomous, these stability analysis tools are limited in their scope and application. Initial results did show that the stabilizing methods used were able to ensure "system energy" was decreasing. Unfortunately, failure to find the critical points for more complex problems limited the application of these tools.

A run time study was conducted to determine the speedup that the substitution of specific volume into the governing equations had on the run time of the ROM. It was found that the zeta-ROM attained a speed up of greater than four orders of magnitude compared to the FOM and to a ROM written in conservative variables. The zeta-ROM ran at an average of 49% of a theoretical upper limit of speedup against the FOM. Improvements to the speedup can be attained by reducing the time it takes to set up the ODE system of the ROM, for example, the zeta-ROM can be sped up by reducing the computational time of reading in files, calculating derivatives, and pre computing the coefficients of the ODEs.

## 8.1 Future Work

While the zeta-ROM is capable of solving for both inviscid and laminar flows for stationary reference frames, it does not capture laminar flows for rotating reference frames and turbulent flows. Equations (3.7) provide the necessary foundation to update the zeta-ROM for laminar flows with a rotating reference frame about the $x$-axis. Section B provides the Spalart-Allmaras turbulence model modified to reduce its nonlinearity such that it can be used effectively with a POD-based reduced-order model. In this case, "effectively" refers to the ability to precompute the inner products or coefficients of the ODE system of the ROM.

In the course of this dissertation, two interpolation methods were used to produce solutions for

off-reference conditions. These interpolation methods interpolated only a single design variable, generally an amplitude or frequency of motion. Interpolating such design variables produced accurate results. One issue, though, is that these interpolation methods may not adequately capture the effect that changes of geometric design variables may have on the solution.

The POD method uses an index-based relationship between grid points. This relationship means that when the computational mesh is deformed, the deformation is not communicated to the POD modes unless the deformation is topologically consistent. In the same way, if the geometric design variables change the computational mesh in a topologically inconsistent manner, the POD modes may not adequately capture the change itself. Even if a proper interpolation method is used, the predicted solution may be inaccurate.

To include the effect of the geometric design variables in the POD modes, it may be possible to instead use the dynamic basis and dynamic average augmented with information on the geometric design variables. Here, the selection of the gamma parameters, $\tilde{\gamma}_i$, are important. For unsteady flows, $\tilde{\gamma}_i$ were used to describe the primary form of motion. For example, a sine or cosine pair describing the motion in time. In addition to this definition, the gamma parameters may also be utilized to bring in information of the geometric design variables into the POD basis. To capture the geometric design variables, the gamma parameters can be written as $\tilde{\gamma}_i = \mathcal{F}_i(\boldsymbol{\alpha}(\mathbf{x}))$ where the gamma parameters are functionals of the geometric design variables, $\boldsymbol{\alpha}$, which may be functions of space, $\mathbf{x}$.

The selection of the functional is very important for the accuracy and computational expense of this method, and can be flexible for the type of problem. For example, consider a linear model of $\tilde{\gamma}_i$

$$\tilde{\gamma}_i = \beta_0 + \beta_1\alpha_1 + \beta_2\alpha_2 + ... + \beta_{n_{\text{var}}}\alpha_{n_{\text{var}}}$$

where $\beta_i$ are coefficients in this model. The model would require $n_{\text{var}} + 1$ full-order model solutions to approximate $\tilde{\gamma}_i$. Depending on the complexity of the design space, this necessary computational expense would still be much cheaper than using the FOM for design. For example, the zeta-ROM

ran four orders of magnitude faster than the FOM. Let us say the optimization procedure takes 10 iteration steps, the ROM takes 1 second to run, and the FOM takes 10,000 seconds to run. Then, the total time of the optimization procedure if using the FOM is 100,000 seconds not including computations to determine gradients. For the optimization procedure using the ROM to run faster than the optimization using the FOM, the number of design variables would be limited to 8. At 8 design variables, the cost of the optimization procedure would be 9 FOM runs, 90,000 seconds, plus 10 ROM runs, 10 seconds, for a total of 90,010 seconds or $1.11\times$ faster than the optimization using the FOM. If one includes multi start into the optimization procedure, the speedup increases by $1.11 \times n_{\text{ms}}$ where $n_{\text{ms}}$ is the number of multi starts. Obviously, the speedup increase would be greater if the cost to calculate the gradients is included.

# REFERENCES


[1] H. M. Park and M. W. Lee, "An efficient method of solving the Navier-Stokes equations for flow control," *International Journal for Numerical Methods in Engineering*, vol. 41, no. 6, pp. 1133–1151, 1998.

[2] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis, "Cfd vision 2030 study: A path to revolutionary computational aerosciences," *NASA/CR-2014-21878*, pp. 1–73, March 2014.

[3] K. Karhunen, "Zur Spektraltheorie Stochasticher Prozesse," *Annales Academiae Scientiarum Fennicae Series A*, vol. 1, p. 34, 1946.

[4] M. Loève, "Functions aléatoire de second order," *Comptes Rendus de l'Académie des Sciences, Paris*, p. 220, 1945.

[5] A. Iollo, S. Lanteri, and J. A. Désidéri, "Stability properties of POD-Galerkin approximations for the compressible Navier-Stokes equations," *Theoretical and Computational Fluid Dynamics*, vol. 13, no. 6, pp. 377–396, 2000.

[6] I. Kalashnikova, S. Arunajatesan, M. F. Barone, B. G. van Bloemen Waanders, and J. A. Fike, "Reduced order modeling for prediction and control of large-scale systems.," tech. rep., Sandia National Laboratories, May 2014.

[7] J. Lumley, "The structure of inhomogenous turbulence," in *Proceedings of the International Colloquium in the Fine Scale Structure of the Atmosphere and its Influence on Radio Wave Propagation* (A. Yaglom, ed.), Dokl Akad Nauk SSSR, 1967.

[8] B. Epureanu, E. Dowell, and K. Hall, "Reduced-order models of unsteady transonic viscous flows in turbomachinery," *Journal of Fluids and Structures*, vol. 14, no. 8, pp. 1215 – 1234, 2000.

[9] P. G. A. Cizmas and A. Palacios, "Proper orthogonal decomposition of turbine rotor-stator interaction," *AIAA Journal of Propulsion and Power*, vol. 19, pp. 268–281, March-April 2003.

[10] K. Nagarajan, S. Singha, L. Cordier, and C. Airiau, "Open-loop control of cavity noise using

Proper Orthogonal Decomposition Reduced-Order Model," *ASME J. Fluids Eng.*, vol. 160, pp. 1–13, 2017.

[11] T. Yuan, P. G. Cizmas, and T. O'Brien, "A reduced-order model for a bubbling fluidized bed based on proper orthogonal decomposition," *Computers and Chemical Engineering*, vol. 30, no. 2, pp. 243–259, 2005.

[12] T. A. Brenner, R. L. Fontenot, P. G. Cizmas, T. J. O'Brien, and R. W. Breault, "Augmented proper orthogonal decomposition for problems with moving discontinuities," *Powder Technology*, vol. 203, no. 1, pp. 78–85, 2010.

[13] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, and C. Wu, "Proper orthogonal decomposition and its applications–part i: Theory," *Journal of Sound and Vibration*, vol. 252, no. 3, pp. 527 – 544, 2002.

[14] L. S. Ramos, K. R. Beebe, W. P. Carey, E. Sanchez, B. C. Erickson, B. E. Wilson, L. E. Wangen, and B. R. Kowalski, "Chemometrics," *Analytical Chemistry*, vol. 58, pp. 294–315, apr 1986.

[15] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[16] O. Alter, P. O. Brown, and D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," *Proceedings of the National Academy of Sciences*, vol. 97, no. 18, pp. 10101–10106, 2000.

[17] J. Mandel, "Use of the singular value decomposition in regression analysis," *The American Statistician*, vol. 36, no. 1, pp. 15–24, 1982.

[18] C. W. Rowley, "Model reduction for fluids using balanced proper orthogonal decomposition," *Int. J. Bifurcation and Chaos*, vol. 15, no. 3, pp. 997–1013, 2005.

[19] N. Aubry, "On the hidden beauty of the proper orthogonal decomposition," *Theoretical and Computational Fluid Dynamics*, vol. 2, no. 5-6, pp. 339–352, 1991.

[20] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of*

*Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[21] D. J. Lucia, P. S. Beran, and W. A. Silva, "Reduced-order modeling: New approaches for computational physics," *Progress in Aerospace Sciences*, vol. 40, no. 1-2, pp. 51–117, 2004.

[22] E. H. Dowell, K. Hall, J. Thomas, R. Florea, B. Epureanu, and J. Heeg, "Reduced order models in unsteady aerodynamics," *40th AIAA/ASME/ASCE/AHS/ASC Struct., Struct. Dyn., and Mater. Conf. AIAA-99-1261*, 1999.

[23] 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*, vol. 228, no. 6, pp. 1932–1946, 2009.

[24] J. S. Anttonen, *Techniques for Reduced Order Modeling of Aeroelastic Structures with Deforming Grids*. PhD thesis, Air Force Institute of Technology, 10 2001.

[25] J. Anttonen, P. King, and P. Beran, "Pod-based reduced-order models with deforming grids," *Mathematical and Computer Modelling*, vol. 38, no. 1, pp. 41 – 62, 2003.

[26] J. Anttonen, P. King, and P. Beran, "Applications of multi-pod to a pitching and plunging airfoil," *Mathematical and Computer Modelling*, vol. 42, no. 3, pp. 245 – 259, 2005.

[27] G. C. Lewin and H. Haj-Hariri, "Reduced-order modeling of a heaving airfoil," *AIAA Journal*, vol. 43, no. 2, pp. 270–283, 2005.

[28] B. A. Freno, *Reduced-Order Models for Computational Aeroelasticity*. PhD thesis, Texas A&M University, 12 2013.

[29] B. A. Freno, N. R. Matula, R. L. Fontenot, and P. G. A. Cizmas, "The use of dynamic basis functions in proper orthogonal decomposition," *Journal of Fluids and Structures*, vol. 54, pp. 332–360, Apr. 2015.

[30] D. Rempfer, "On Low-Dimensional Galerkin Models for Fluid Flow," *Theoretical and Computational Fluid Dynamics*, vol. 14, pp. 75–88, Jun 2000.

[31] B. N. Bond and L. Daniel, "Guaranteed stable projection-based model reduction for indefinite and unstable linear systems," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, pp. 728–735, 2008.

[32] D. Amsallem and C. Farhat, "Stabilization of projection-based reduced-order models," *International Journal for Numerical Methods in Engineering*, vol. 91, no. 4, pp. 358–377, 2012.

[33] I. Kalashnikova, B. van Bloemen Waanders, S. Arunajatesan, and M. Barone, "Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment," *Computer Methods in Applied Mechanics and Engineering*, vol. 272, pp. 251–270, 2014.

[34] M. Tomas-Rodriguez and S. P. Banks, "An iterative approach to eigenvalue assignment for nonlinear systems," *International Journal of Control*, vol. 86, no. 5, pp. 883–892, 2013.

[35] I. Kalashnikova and M. F. Barone, "On the stability and convergence of a Galerkin reduced order model (ROM) of compressible flow with solid wall and far-field boundary treatment," *International Journal for Numerical Methods in Engineering*, vol. 83, no. 10, pp. 1345–1375, 2010.

[36] D. J. Lucia and P. S. Beran, "Projection methods for reduced order models of compressible flows," *Journal of Computational Physics*, vol. 188, no. 1, pp. 252–280, 2003.

[37] S. Sirisup and G. E. Karniadakis, "A spectral viscosity method for correcting the long-term behavior of POD models," *Journal of Computational Physics*, vol. 194, no. 1, pp. 92–116, 2004.

[38] J. Borggaard, T. Iliescu, and Z. Wang, "Artificial viscosity proper orthogonal decomposition," *Mathematical and Computer Modelling*, vol. 53, no. 1-2, pp. 269–279, 2011.

[39] E. H. Krath, J. A. Felderhoff, N. R. Matula, P. G. A. Cizmas, and D. A. Johnston, "A reduced-order model for compressible flows based on an efficient proper orthogonal decomposition method," in *Proceedings of the 15th International Symposium on Unsteady Aerodynamics, Aeroacoustics & Aeroelasticity of Turbomachines*, no. ISUAAAT15-012, (University of Oxford, UK), September 2018.

[40] E. H. Krath, F. L. Carpenter, P. G. A. Cizmas, and D. A. Johnston, "Reduced-Order Model of Unsteady Flows in a Turbomachinery Fan Modeled using a Novel Proper Orthogonal

Decomposition," in *Proceedings of ASME Turbo Expo 2019: Turbomachinery Technical Conference and Exposition*, 2019.

[41] E. H. Krath, F. L. Carpenter, P. G. Cizmas, and D. A. Johnston, "An efficient proper orthogonal decomposition based reduced-order model for compressible flows," *Journal of Computational Physics*, 2020.

[42] J. Hesthaven and D. Gottlieb, "A Stable Penalty Method for the Compressible Navier-Stokes Equations: I. Open Boundary Conditions," *SIAM Journal of Scientific Computing*, vol. 17, no. 3, pp. 579–612, 1996.

[43] I. Kalashnikova and M. F. Barone, "Efficient non-linear proper orthogonal decomposition/-Galerkin reduced order models with stable penalty enforcement of boundary conditions," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 11, pp. 1337–1362, 2012.

[44] E. J. Doedel, "Auto: A program for the automatic bifurcation analysis of autonomous systems," in *Proceedings of the 10th Manitoba Conference on Numerical Mathematics and Computations*, (University of Manitoba, Winnipeg, Canada), pp. 265–284, 1981.

[45] D. Amsallem and C. Farhat, "Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity," *AIAA Journal*, vol. 46, no. 7, pp. 1803–1813, 2008.

[46] B. A. Freno, T. A. Brenner, and P. G. Cizmas, "Using proper orthogonal decomposition to model off-reference flow conditions," *International Journal of Non-Linear Mechanics*, vol. 54, pp. 76–84, 2013.

[47] M. Fahl and E. W. Sachs, *Reduced Order Modelling Approaches to PDE-Constrained Optimization Based on Proper Orthogonal Decomposition*, pp. 268–280. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[48] D. My-Ha, K. Lim, B. Khoo, and K. Willcox, "Real-time optimization using proper orthogonal decomposition: Free surface shape prediction due to underwater bubble dynamics," *Computers & Fluids*, vol. 36, no. 3, pp. 499 – 512, 2007.

[49] D. J. J. Toal, N. W. Bressloff, A. J. Keane, and C. M. E. Holden, "Geometric filtration using

proper orthogonal decomposition for aerodynamic design optimization," *AIAA Journal*, vol. 48, no. 5, pp. 916–928, 2010.

[50] P. LeGresley and J. Alonso, "Investigation of non-linear projection for pod based reduced order models for aerodynamics," in *39th Aerospace Sciences Meeting and Exhibit*.

[51] T. Bui-Thanh, M. Damodaran, and K. Willcox, "Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition," *AIAA Journal*, vol. 42, no. 8, pp. 1505–1516, 2004.

[52] P. G. A. Cizmas, A. Palacios, T. O'Brien, and M. Syamlal, "Proper-orthogonal decomposition of spatio-temporal patterns in fluidized beds," *Chemical Engineering Science*, vol. 58, pp. 4417–4427, October 2003.

[53] P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 1996.

[54] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis*. New York: Interscience Publishers, Inc., 1964.

[55] R. Pinnau, "Model reduction via proper orthogonal decomposition," in *Model Order Reduction: Theory, Research Aspects and Applications* (W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, eds.), Mathematics in Industry, pp. 95–109, Springer, 2008.

[56] L. Sirovich, "Turbulence and the dynamics of coherent structures, I-III," *Quarterly of Applied Mathematics*, vol. 45, no. 3, pp. 561–590, 1987.

[57] P. Roe, "Characteristic-Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics*, vol. 18, no. 1, pp. 337–365, 1986.

[58] J. D. Anderson, *Computational Fluid Dynamics: The Basics with Applications*. McGrawhill Inc, 1995.

[59] Z.-X. Han and P. G. Cizmas, "A CFD Method for Axial Thrust Load Prediction of Centrifugal Compressors," *International Journal of Turbo and Jet Engines*, vol. 20, no. 1, pp. 1–16, 2003.

[60] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[61] A. Harten, J. M. Hyman, and P. D. Lax, "On finite-difference approximations and entropy conditions for shocks," *Communications on Pure and Applied Mathematics*, vol. 29, no. 3, pp. 297–322, 1976.

[62] T. J. Barth, "Aspects of unstructured grids and finite-volume solvers for the euler and navier-stokes equations," in *Special Course on Unstructured Grid Methods for Advection Dominated Flows*, vol. R-787, AGARD, 1992.

[63] S. Yoon and D. Kwak, "Implicit navier-stokes solver for three-dimensional compressible flows," *AIAA Journal*, vol. 30, pp. 2653–2659, November 1992.

[64] P. D. Thomas and C. K. Lombard, "Geometric conservation law and its application to flow computations on moving grids," *AIAA Journal*, vol. 17, pp. 1030–1037, October 1979.

[65] T. A. Brenner, *Practical Aspects of the Implementation of Reduced-Order Models Based on Proper Orthogonal Decomposition*. PhD thesis, Texas A&M University, 5 2011.

[66] "A reduced-order model for heat transfer in multiphase flow and practical aspects of the proper orthogonal decomposition," *Computers & Chemical Engineering*, vol. 43, pp. 68–80, August 2012.

[67] J. Burkardt, M. Gunzburger, and H. C. Lee, "POD and CVT-based reduced-order modeling of Navier-Stokes flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 1-3, pp. 337–355, 2006.

[68] A. M. Liapunov and A. T. Fuller, *The general problem of the stability of motion. A.M. Lyapunov ; translated and edited by A.T. Fuller ; with a biography of Lyapunov by V.I. Smirnov and a bibliography of Lyapunov's works by J.F. Barrett.* London ; Washington, DC : Tayor & Francis, 1992., 1992.

[69] J.-J. E. Slotine and W. Li, *Applied nonlinear control. Jean-Jacques E. Slotine, Weiping Li.* Englewood Cliffs, N.J. : Prentice Hall, [1991], 1991.

[70] D. Funaro and D. Gottlieb, "Convergence results for pseudospectral by a penalty-type boundary treatment," *Mathematics of Computation*, vol. 57, no. 196, pp. 585–596, 1991.

[71] D. Amsallem, J. Cortial, and C. Farhat, "Towards real-time computational-fluid-dynamics-

based aeroelastic computations using a database of reduced-order information," *AIAA Journal*, vol. 48, no. 9, pp. 2029–2037, 2010.

[72] A. C. Hindmarsh, "ODEPACK, A Systematized Collection of ODE Solvers," *Scientific Computing*, vol. 1, pp. 55–64, 1983.

[73] J. Blazek, *Computational Fluid Dynamics: Principles and Applications*. Elsevier, second ed., 2005.

[74] M.-S. Liou, "A generalized procedure for constructing an upwind-based TVD scheme," in *25th AIAA Aerospace Sciences Meeting*, Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics, mar 1987.

[75] D. C. Urasek, W. T. Gorrell, and W. S. Cunnan, "Performance of Two-Stage Fan Having Low-Aspect Ratio, First-Stage Rotor Blading," in *NASA Technical Paper*, 1979.

[76] F. L. Carpenter IV, *Practical Aspects of Computational Fluid Dynamics for Turbomachinery*. PhD thesis, Texas A&M University, College Station, Texas, August 2016.

[77] J. VERDON, *Linearized unsteady aerodynamics for turbomachinery aeroelastic applications*. 1990.

[78] T. H. Fransson and J. M. Verdon, "Updated report on standard configurations for the determination of unsteady flow through vibrating axial-flow turbomachine-cascades," *Proceedings of the 6th International Conference on Aeroelasticity in Turbomachines*, 1991.

[79] A. de Boer, M. van der Schoot, and H. Bijl, "Mesh deformation based on radial basis function interpolation," *Computers & Structures*, vol. 85, no. 11, pp. 784 – 795, 2007. Fourth MIT Conference on Computational Fluid and Solid Mechanics.

[80] T. H. Fransson and J. M. Verdon, "Standard configurations for unsteady flow through vibrating axial-flow turbomachine cascades," in *Unsteady Aerodynamics, Aeroacoutics and Aeroelasticity of Turbomachines and Propellers* (H. M. Atassi, ed.), (New York), pp. 859–889, Springer, 1993.

[81] S. Allmaras, F. Johnson, and P. Spalart, "Modifications and clarifications for the implementation of the spalart-allmaras turbulence model," *Seventh International Conference on*

*Computational Fluid Dynamics (ICCFD7)*, pp. 1–11, 01 2012.

# APPENDIX A

## ISENTROPIC BOUNDARY CONDITION FOR SPECIFIC VOLUME

An isentropic boundary condition for specific volume at the outlet is formulated using the isentropic relations

$$\frac{p^*}{p} = \left(\frac{\rho^*}{\rho}\right)^\gamma = \left(\frac{\zeta}{\zeta^*}\right)^\gamma$$

with nondimensionalization

$$p' = \frac{p}{q_{ref}} \qquad \rho' = \frac{\rho}{\rho_{ref}} \qquad T' = \frac{T}{T_{ref}}$$

where $(\cdot)'$ denotes a nondimensional term. Here $q_{ref} = \rho_{ref} v_{ref}^2$ and $v_{ref} = \sqrt{\gamma R T_{ref}}$. To begin, the isentropic relations are nondimensionalized.

$$\frac{p^*}{p' q_{ref}} = \left(\frac{\rho^*}{\rho' \rho_{ref}}\right)^\gamma = \left(\frac{\zeta'}{\zeta^* \rho_{ref}}\right)^\gamma$$

At the outlet, the dimensionless pressure, $p'$, is oscillated with function $f(t)$.

$$\frac{p^*}{f(t) q_{ref}} = \left(\frac{\zeta'}{\zeta^* \rho_{ref}}\right)^\gamma$$

Switch the powers and multiply by $\rho_{ref}/\rho^*$ on both sides.

$$\begin{aligned}
\left(\frac{p^*}{f(t) q_{ref}}\right)^{\frac{1}{\gamma}} &= \frac{\zeta'}{\zeta^* \rho_{ref}} \\
\frac{\rho_{ref}}{\rho^*}\left(\frac{p^*}{f(t) q_{ref}}\right)^{\frac{1}{\gamma}} &= \frac{\rho_{ref}}{\rho^*}\frac{\zeta'}{\zeta^* \rho_{ref}} \\
\frac{\rho_{ref}}{\rho^*}\left(\frac{p^*}{f(t) \rho_{ref} v_{ref}^2}\right)^{\frac{1}{\gamma}} &= \frac{\zeta'}{\zeta^* \rho^*} \\
\frac{\rho_{ref}}{\rho^*}\left(\frac{p^*}{f(t) \rho_{ref} \gamma R T_{ref}}\right)^{\frac{1}{\gamma}} &= \zeta'
\end{aligned}$$

$$\frac{1}{\rho^*}\left(\frac{p^*\rho_{ref}^{\gamma-1}}{f(t)\gamma RT_{ref}}\right)^{\frac{1}{\gamma}} = \zeta'$$

Here the equations for $q_{ref}$ and $v_{ref}$ were substituted in. The equation is simplified even further by assuming that the flow behaves as an ideal gas.

$$\frac{1}{\rho^*}\left(\frac{\rho^* RT^*\rho_{ref}^{\gamma-1}}{f(t)\gamma RT_{ref}}\right)^{\frac{1}{\gamma}} = \zeta'$$

$$\zeta' = \left(\frac{T^*\rho_{ref}^{\gamma-1}}{f(t)\gamma T_{ref}(\rho^*)^{\gamma-1}}\right)^{\frac{1}{\gamma}} \tag{A.1}$$

This is further simplified if the reference values correspond to the total values.

$$\zeta' = (\gamma f(t))^{-\frac{1}{\gamma}} \tag{A.2}$$

# APPENDIX B

## REDUCED-ORDER SPALART ALLMARAS EQUATIONS

The S-A model for turbulent flows models the Reynolds stresses using the Boussinesq eddy viscosity assumption. Here, the eddy viscosity is defined as $\nu_t = \tilde{\nu} f_{\nu_1}$ where $\tilde{\nu}$ is the S-A working variable and

$$f_{\nu_1} = \frac{\chi^3}{\chi^3 + c_{\nu_1}^3} \tag{B.1}$$

where $\chi = \tilde{\nu}/\nu$ and $\nu$ is the kinematic viscosity. The S-A working variable has a transport equation of the form

$$\frac{D\tilde{\nu}}{Dt} = \mathcal{P} - \mathcal{D} + \mathcal{T} + \frac{1}{\sigma}\left(\nabla \cdot \left(\left(\nu + \tilde{\nu}\right)\nabla\tilde{\nu}\right) + c_{b_2}\left(\nabla\tilde{\nu}\right)^2\right) \tag{B.2}$$

where $\sigma, c_{b_2}$ are constants, $\nu$ is the kinematic viscosity, and $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{T}$ are the production, dissipation, and trip terms respectively. The production term is

$$\mathcal{P} = c_{b_1}\left(1 - f_{t_2}\right)\tilde{S}\tilde{\nu} \tag{B.3}$$

where $c_{b_1}$ is a constant, $\tilde{S}$ is the modified vorticity, and $f_{t_2}$ is the laminar suppression term,

$$f_{t_2} = c_{t_3} e^{-c_{t_4}\chi^2} \tag{B.4}$$

where $c_{t_3}$ and $c_{t_4}$ are constants. Physically, the modified vorticity should always be negative with values that never fall below $0.3S$. A modified form of $\tilde{S}$ was developed by Spalart and Allmaras [81]

$$\tilde{S} = \begin{cases} S + \bar{S} & \bar{S} \geq -c_{\nu_2} S \\ S + \frac{S\left(c_{\nu_2}^2 S + c_{\nu_3}\bar{S}\right)}{\left(c_{\nu_3} - 2c_{\nu_2}\right)S - \bar{S}} & \bar{S} < -c_{\nu_2} S \end{cases}, \quad \bar{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{\nu_2} \tag{B.5}$$

where $\kappa, c_{\nu_2}, c_{\nu_3}$ are constants, $S$ is the magnitude of the vorticity,

$$S = \sqrt{\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)^2}$$

$d$ is the distance closest to the wall, $f_{t_2} = c_{t_3} e^{-c_{t_4}\chi^2}$ where $c_{t_3}, c_{t_4}$ are constants, and

$$f_{\nu_2} = 1 - \frac{\chi}{1 + \chi f_{\nu_1}}. \tag{B.6}$$

The dissipation term is

$$\mathcal{D} = \left(c_{w_1} f_w - \frac{c_{b_1}}{\kappa^2} f_{t_2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 \tag{B.7}$$

$$f_w = g\left(\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right)^{1/6}, \quad g = r + c_{w_2}\left(r^6 - r\right), \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, r_{\text{lim}}\right).$$

where $c_{w_1}, c_{w_2}, c_{w_3}$, and $r_{\text{lim}}$ are constants. The trip term is

$$\mathcal{T} = f_{t_1}\left(\nabla u\right)^2 \tag{B.8}$$

$$f_{t_1} = c_{t_1} g_t e^{-c_{t_2}\frac{\omega_t}{\Delta u^2}\left(d^2 + g_t^2 d_t^2\right)}, \; g_t = \min\left(0.1, \frac{\Delta u}{\omega_t \Delta x}\right)$$

where $d_t$ is the distance to the trip point, $\omega_t$ is the vorticity of the trip, $\Delta u$ is the difference in velocity relative to the trip point, $\Delta x$ is the streamwise grid spacing at the trip, and $c_{t_1}, c_{t_2}$ are constants. Herein, the trip term $\mathcal{T}$ is neglected. All constants and their values are given in Table B.1.

## B.1 Reduced-Order Model

To develop a POD-based reduced-order model, the equations need to be converted into a form where the projection along the optimal subspace will not be time-dependent. To do this, the binomial theorem, multinomial theorem, which is a special case of the binomial theorem, and a power series expansion of the exponential function were employed.

$$\text{Binomial Theorem} : (x + y)^n = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} x^{n-k} y^k \tag{B.9}$$

| Constants | Values |
|---|---|
| $c_{b_1}$ | 0.1355 |
| $c_{b_2}$ | 0.622 |
| $c_{w_1}$ | $\frac{c_{b_1}}{\kappa^2} + \frac{1+c_{b_2}}{\sigma}$ |
| $c_{w_2}$ | 0.3 |
| $c_{w_3}$ | 2 |
| $c_{\nu_1}$ | 7.1 |
| $c_{\nu_2}$ | 0.7 |
| $c_{\nu_3}$ | 0.9 |
| $c_{t_1}$ | 1 |
| $c_{t_2}$ | 2 |
| $c_{t_3}$ | 1.2 |
| $c_{t_4}$ | 0.5 |
| $\sigma$ | 2/3 |
| $\kappa$ | 0.41 |
| $r_{\text{lim}}$ | 10 |

Table B.1: Constants of the S-A Equation

$$\text{Multinomial Theorem} : (x_1 + x_2 + \cdots + x_m)^n = \sum_{k_1+k_2+\cdots+k_m=n} \frac{n!}{k_1!k_2!\cdots k_m!} \prod_{t=1}^{m} x_t^{k_t} \tag{B.10}$$

$$\text{Power Series of } e^x : e^x = \sum_{k=0}^{\infty} \frac{x^n}{n!} \tag{B.11}$$

For example, the multinomial theorem is applied directly to the POD approximation in the case where $\tilde{\nu}$ is raised to a power $n$.

$$\begin{aligned}
(\tilde{\nu})^n &= \left( \sum_{i=0}^{n^{\tilde{\nu}}} a_i^{\tilde{\nu}}(t)\phi_i^{\tilde{\nu}}(x) \right)^n \\
&= \left( a_0^{\tilde{\nu}}(t)\phi_0^{\tilde{\nu}} + a_1^{\tilde{\nu}}(t)\phi_1^{\tilde{\nu}} + \cdots + a_{n^{\tilde{\nu}}}^{\tilde{\nu}}(t)\phi_{n^{\tilde{\nu}}}^{\tilde{\nu}} \right)^n \\
&= \sum_{p_0+p_1+\cdots+p_{n^{\tilde{\nu}}}=n} \frac{n!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left( a_i^{\tilde{\nu}}(t)\phi_i^{\tilde{\nu}}(x) \right)^{p_i} \\
&= \sum_{p_0+p_1+\cdots+p_{n^{\tilde{\nu}}}=n} \frac{n!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left( a_i^{\tilde{\nu}}(t) \right)^{p_i} \left( \phi_i^{\tilde{\nu}}(x) \right)^{p_i}
\end{aligned}$$

To complete separating the spatial-dependent variable, $\phi$, from any time-dependent variables, the product operator property

$$\prod_{i=1}^{I} x_i y_i = \left(\prod_{i=1}^{I} x_i\right)\left(\prod_{i=1}^{I} y_i\right)$$

is used. Therefore, the inner product is of the form $\left(\prod_{i=0}^{n^{\tilde{\nu}}} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i}, \phi_k^{\tilde{\nu}}(x)\right)$, which can be pre-calculated for any value of $p_i$. There may be memory issues if $n^{\tilde{\nu}}$ is large.

The S-A equations were approximated using these theorems and series expansions. The following sections provide the approximations to the production and dissipation terms and to eddy viscosity.

### B.1.1 Production, $\mathcal{P}$

The production term in Eq. (B.3) is repeated here for convenience.

$$\mathcal{P} = c_{b_1}\left(1 - f_{t_2}\right)\tilde{S}\tilde{\nu}$$

One nonlinear function of $\tilde{\nu}$ appears in the equation: the modified vorticity $\tilde{S}$. To prevent negative values of $\tilde{S}$, a new formulation based off of two inequalities was presented by Spalart and Allmaras [81] and was given in Eq. (B.5). The inequality expressed in Eq. (B.5) needs to be approximated such that it can be projected along the POD subspace without producing time-dependent inner products. The function $f_{t_2}$ is approximated as

$$\begin{aligned} f_{t_2} &= c_{t_3} e^{-c_{t_4}\chi^2} \\ &= c_{t_3} \sum_{k=0}^{\infty} \frac{\left(-\frac{c_{t_4}}{\nu^2}\right)^k}{k!} \tilde{\nu}^{2k} \\ &= c_{t_3} \sum_{k=0}^{\infty} \frac{\left(-\frac{c_{t_4}}{\nu^2}\right)^k}{k!} \sum_{p_0+p_1+\cdots+p_{n^{\tilde{\nu}}}=2k} \frac{(2k)!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left(a_i^{\tilde{\nu}}(t)\right)^{p_i} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i}. \end{aligned} \tag{B.12}$$

The function $f_{\nu_2}$ is approximated as

$$\begin{aligned} f_{\nu_2} &= 1 - \frac{\chi}{1+\chi f_{\nu_1}} \\ &= 1 - \sum_{k=0}^{\infty} \frac{(-1)^k}{(c_{\nu_1}\nu)^{3k}} \sum_{m=0}^{k} \frac{k!}{m!(k-m)!} \left( \tilde{\nu}^{3k+m+4} + \frac{1}{c_{\nu_1}^2 \nu} \tilde{\nu}^{3k+m+1} \right) \\ &= 1 - C_{\nu_2} \end{aligned} \tag{B.13}$$

where

$$\begin{aligned} C_{\nu_2} = \sum_{k=0}^{\infty} \frac{(-1)^k}{(c_{\nu_1}\nu)^{3k}} \sum_{m=0}^{k} \frac{k!}{m!(k-m)!} \Bigg[ & \sum_{p_0+p_1+\cdots+p_{n^{\tilde{\nu}}}=3k+m+4} \frac{(3k+m+4)!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left(a_i^{\tilde{\nu}}(t)\right)^{p_i} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i} \\ & - \frac{1}{c_{\nu_1}^2 \nu} \sum_{p_0+p_1+\cdots+p_{n^{\tilde{\nu}}}=3k+m+1} \frac{(3k+m+1)!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left(a_i^{\tilde{\nu}}(t)\right)^{p_i} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i} \Bigg]. \end{aligned}$$

The term $\bar{S}$ can now be written in terms of this approximation.

$$\begin{aligned} \bar{S} &= \frac{\tilde{\nu}}{\kappa^2 d^2} f_{\nu_2} \\ &= \frac{\tilde{\nu}}{\kappa^2 d^2} \left( 1 - \sum_{k=0}^{\infty} \frac{(-1)^k}{(c_{\nu_1}\nu)^{3k}} \sum_{m=0}^{k} \frac{k!}{m!(k-m)!} \left( \tilde{\nu}^{3k+m+4} + \frac{1}{c_{\nu_1}^2 \nu} \tilde{\nu}^{3k+m+1} \right) \right) \\ &= \frac{\tilde{\nu}}{\kappa^2 d^2} - \frac{1}{\kappa^2 d^2} \sum_{k=0}^{\infty} \frac{(-1)^k}{(c_{\nu_1}\nu)^{3k}} \sum_{m=0}^{k} \frac{k!}{m!(k-m)!} \left( \tilde{\nu}^{3k+m+5} + \frac{1}{c_{\nu_1}^2 \nu} \tilde{\nu}^{3k+m+2} \right) \end{aligned}$$

### B.1.2 Dissipation, $\mathcal{D}$

The dissipation term in Eq. (B.7) is repeated here for convenience.

$$\mathcal{D} = \left( c_{w_1} f_w - \frac{c_{b_1}}{\kappa^2} f_{t_2} \right) \left( \frac{\tilde{\nu}}{d} \right)^2$$

Two nonlinear functions of $\tilde{\nu}$ appear in the equation: the functions $f_w$ and $f_{t_2}$. The approximation for $f_{t_2}$ is

$$\begin{aligned} f_{t_2} &= c_{t_3} e^{-c_{t_4}\chi^2} \\ &= c_{t_3} \sum_{k=0}^{\infty} \frac{\left(-\frac{c_{t_4}}{\nu^2}\right)^k}{k!} \tilde{\nu}^{2k} \\ &= c_{t_3} \sum_{k=0}^{\infty} \frac{\left(-\frac{c_{t_4}}{\nu^2}\right)^k}{k!} \sum_{p_0+p_1+\cdots+p_{n^{\tilde{\nu}}}=2k} \frac{(2k)!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left(a_i^{\tilde{\nu}}(t)\right)^{p_i} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i}. \end{aligned} \tag{B.14}$$

### B.1.3 Eddy Viscosity, $\nu_t$

The eddy viscosity contains one nonlinear function of $\tilde{\nu}$: the function $f_{\nu_1}$. This function is approximated as

$$\begin{aligned} f_{\nu_1} &= \frac{\chi^3}{\chi^3 + c_{\nu_1}^3} \\ &= \frac{1}{\left(c_{\nu_1}\nu\right)^3} \sum_{k=0}^{\infty} \frac{(-1)^k}{\left(c_{\nu_1}\nu\right)^{3k}} \tilde{\nu}^{3(1+k)} \\ &= \frac{1}{\left(c_{\nu_1}\nu\right)^3} \sum_{k=0}^{\infty} \frac{(-1)^k}{\left(c_{\nu_1}\nu\right)^{3k}} \sum_{p_0+p_1+\cdots p_{n^{\tilde{\nu}}}=3(1+k)} \frac{(3(1+k))!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left(a_i^{\tilde{\nu}}(t)\right)^{p_i} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i} \end{aligned} \tag{B.15}$$

where the final approximation for the eddy viscosity is

$$\begin{aligned} \nu_t &= \tilde{\nu} f_{\nu_1} \\ &= \tilde{\nu} \left( \frac{1}{\left(c_{\nu_1}\nu\right)^3} \sum_{k=0}^{\infty} \frac{(-1)^k}{\left(c_{\nu_1}\nu\right)^{3k}} \tilde{\nu}^{3(1+k)} \right) \\ &= \frac{1}{\left(c_{\nu_1}\nu\right)^3} \sum_{k=0}^{\infty} \frac{(-1)^k}{\left(c_{\nu_1}\nu\right)^{3k}} \tilde{\nu}^{4+3k} \\ &= \frac{1}{\left(c_{\nu_1}\nu\right)^3} \sum_{k=0}^{\infty} \frac{(-1)^k}{\left(c_{\nu_1}\nu\right)^{3k}} \sum_{p_0+p_1+\cdots p_{n^{\tilde{\nu}}}=4+3k} \frac{(4+3k)!}{p_0!p_1!\cdots p_{n^{\tilde{\nu}}}!} \prod_{i=0}^{n^{\tilde{\nu}}} \left(a_i^{\tilde{\nu}}(t)\right)^{p_i} \left(\phi_i^{\tilde{\nu}}(x)\right)^{p_i}. \end{aligned}$$

APPENDIX C

ZETA-VARIABLE POD-ROM USER'S MANUAL

This appendix contains all necessary documentation for running the zeta-variable POD-ROM code. To access the software, please email cizmas@tamu.edu

## C.1 Software Installation

This section details the installation of several software necessary to run the $\zeta$-ROM .

### C.1.1 `unsromBC` Installation

The `unsromBC` software and all auxiliary files are contained in the `zetaunsromBC.tgz` file. It is recommended to move this file to its own directory on the user's system. To untar the file type, `cd` to the `.tgz` file directory and type

```
tar -xf zetaunsromBC.tgz
```

in the terminal. Six `.tgz` files will be extracted. These files will also need to be extracted using the `tar` command. These files contain six folders: the `BC_interpolation` directory containing the `BC_interpolation` MATLAB software, the `convert` directory containing the `convert` software, the `unsromBC` directory containing the $\zeta$-ROM software along with libraries used, the `bfg` directory containing the `bfg` code to generate basis functions and time coefficients from a set of snapshots, the `grassman` directory containing the `grass` code that interpolates basis functions, and an additional directory containing an auxiliary program `findsurface` that is used for blade deformation. A copy of this manual is also provided.

The `findsurface` software should be compiled in its respective folder. To compile the software type

```
gfortran findsurface.f90 -o findsurface
```

in the terminal. If the user does not have `gfortran` installed, they may use whichever Fortran compiler they prefer.

The BC_interpolation software needs no installation, but requires a working license of MATLAB.

The convert software should be compiled in its respective folder. To compile the software type

```
gfortran convert.f90 -o convert
```

in the terminal. If the user does not have gfortran installed, they may use whichever Fortran compiler they prefer.

The bfg software installation is given in Sec. C.1.2. Information on the grassman software is given in Sec. C.3.

The unsromBC directory contains five folders:

example - example cases that the user may use to practice running the software (see Sec. C.4.2)

inputfiles - contains the input files for the $\zeta$-ROM code

minpack - contains the source files to create the minpack library used for compilation of the $\zeta$-ROM code

odepack - contains the source files to create the odepack library used for compilation of the $\zeta$-ROM code

src - contains the source code for the $\zeta$-ROM code

Before installing the $\zeta$-ROM code, three libraries must be installed: minpack, odepack, and LAPACK. The LAPACK library is not given here, but can be found online or in the documentation for uns3d. The minpack library is already included in its folder and is named minpack.a. The odepack library needs to be created using either a gfortran compiler or f95 compiler. To install simply type make in the terminal.

After the libraries have been installed, the $\zeta$-ROM code can be compiled. Go into the src directory by typing

```
cd src
```

in the terminal. Link in the `minpack` and `odepack` libraries into the diretory by typing

```
ln -s ../minpack/minpack.a .
ln -s ../odepack/f95/opk.a odepack_dpf95.a
```

for `f95` compilers or

```
ln -s ../minpack/minpack.a .
ln -s ../odepack/gfortran/opk.a odepack_dpf90.a
```

for `gfortran` compilers. It is important to note that the $\zeta$-ROM code has been optimized for an `f95` compiler, and may not necessarily work for other compilers.

### C.1.2 `bfg` Installation

The `bfg` code develops the basis functions and time coefficients used by the $\zeta$-ROM code. This code has been modified to accept the outputs of the `convert` code. The installation of the `bfg` software requires the use of the LAPACK library from `www.netlib.org/lapack`. If you do not have these libraries already installed on your computer, you can download them from the `www.netlib.org` web page. Installing the LAPACK libraries correctly is critical to the proper operation of the `bfg` software.

To install the `bfg` code, follow these steps:

1. Untar the file `bfg.tgz`:

   ```
   tar xvfz bfg.tgz
   ```

2. Edit the makefile by commenting out any compiler setup lines that are not related to your compiler. Also, make sure the location of your LAPACK libraries are called out correctly

3. Make the makefile:

   ```
   make
   ```

## C.2 Basis Function Generator (`bfg`)

This section describes how to run the basis function generator (`bfg` ) using terminal line commands. The basis functions are computed using snapshots of the flow field. The reduced-order model (ROM) based on proper orthogonal decomposition (POD) uses the basis functions to compute corresponding time coefficients to reconstruct the solution.

### C.2.1 Running `bfg`

To run the basis function generator, the input file `bfg_input.nml` and the folders `mesh` and `out` are needed. These folders should have been populated with files when the `UNS3D POD` was run. For example, the `out` directory should contain:

`rhosnaps.dat` - $\rho$ state variable

`rhousnaps.dat` - $\rho u$ state variable

`rhovsnaps.dat` - $\rho v$ state variable

`rhowsnaps.dat` - $\rho w$ state variable

`rhoEsnaps.dat` - $\rho E$ state variable

When the `UNS3D POD` code is ran with multiple processors, these snapshot files will additionally contain an underscore and a three character string containing which processor the file was outputted from. For example, for three processors the `rhosnaps.dat` file will be written out as:

`rhosnaps.dat_000`

`rhousnaps.dat_001`

`rhousnaps.dat_002`

Additionally, if the basis function generator is used to produce a dynamic average and/or dynamic basis functions, an additional file, `gammas.dat` is required which should be outputted from the

UNS3D POD code and also stored in out. This file consists of (number of snapshots) rows with columns consisting of the time of the snapshot and $\gamma_1, \ldots, \gamma_d$.

If one is using the zeta-coordinate POD ROM code, then the snapshot files outputted by the UNS3D POD code need to be converted to a format acceptable to the zeta-coordinate POD ROM code. Firstly, if the UNS3D POD code was ran in parallel, the parallel output snapshot files need to be combined into one file. The combineout code will accomplish this. See Appendix C.7 for more details. After combining the snapshot files, the convert code described in Sec. C.6.1 should be used to convert the snapshot files into a format acceptable to the zeta-coordinate POD ROM

To generate the basis functions, switch to the directory in which bfg is to be executed. This directory must contain the input file bfg_input.nml and the snapshot files. It is recommended to make a separate directory from the out directory and link in the snapshot files. To run bfg , the following command statement can be used:

```
bfg snapshot_file basis_file tc_file energy_spectrum
```

The snapshot_file will be the name of the snapshot file outputted by the UNS3D POD code. The basis_file, tc_file, and energy_spectrum files are the names of the files containing the basis functions, time coefficients, and energy spectrum outputted by the bfg code respectively.

Alternately, two executable files titled run_UNS3D and run_zeta are also included in the bfg installation files. The run_UNS3D file contains the execution statements:

```
./bfg rhosnaps.dat  rrm art  r_spect

./bfg rhousnaps.dat rum aut ru_spect

./bfg rhovsnaps.dat rvm avt rv_spect

./bfg rhowsnaps.dat rwm awt rw_spect

./bfg rhoEsnaps.dat rEm aEt rE_spect
```

Here you can see that the `bfg` code needs to be rerun for each snapshot file. The `run_zeta` file contains the execution statements:

```
./bfg z.dat zm zt z_spect

./bfg u.dat um ut u_spect

./bfg v.dat vm vt v_spect

./bfg w.dat wm wt w_spect

./bfg p.dat pm Et p_spect
```

This file produces basis functions and time coefficients for the zeta-coordinate POD ROM, which uses specific volume in place of density in the governing equations.

### C.2.2 `bfg` Input and Output Files

#### *C.2.2.1 `bfg` Input File*

The `bfg` input file, `bfg_input.nml`, consists of the following parameters:

## &bfg_input

| | | |
|---|---|---|
| `nss` | I | The number of snapshots of the flow field. If the `convert` code was used, use the number of snapshots of the converted snapshot files |
| `skip_snaps` | I | The number of snapshots to skip (at the beginning of the sequence of snapshots) when determining the basis functions. This will be zero if the snapshots were already skipped in the `convert` code. |
| `skip_aux` | I | The number of snapshots to skip in the `numsnaps.dat` and `gammas.dat` files |
| `n` | I | The number of nodes |

| | | |
|---|---|---|
| `n_modes` | I | The number of basis functions |
| `dyn_avg` | L | Whether a dynamic average is used |
| `dyn_phi` | L | Whether dynamic basis functions are used |
| `lbfgs` | L | Whether the dynamic basis functions are obtained using the L-BFGS algorithm |
| `n_parameters` | I | The number of parameters used by dynamic average/basis functions |
| `parallel` | L | Whether the snapshots and basis functions are used for the parallel version of `UNS3D POD` (.true.) |
| `ncpu` | I | The number of processors used by the parallel version of `UNS3D POD` |
| `snapshot_dir` | C | Directory containing the snapshots |
| `loc2glob_dir` | C | Directory containing the node distribution across the processors, *i.e.*, the `loc2glob` file |

Many of the parameters needed for the `bfg_input.nml` file have special constraints to their values. The following list shows how to determine the proper values of these parameters:

`nss`
: The number of snapshots can be found by reading the length of the `numsnaps.dat` file

```
cd out/
wc numsnaps.dat
```

The number of snapshots is the first number listed. Take note that if the `convert` code was used, then the number of snapshots of the snapshot files to the number of snapshots of the `gammas.dat` and `numsnaps.dat` files will not match. The `bfg` code can be used to

correct for the difference, but it will not modify the `gammas.dat` and `numsnaps.dat` files.

`skip` — To determine this value, information from the convergence of the full order model must be known. When the number of snapshots is very low skip should be set to 0 or more snapshots should be developed. If the time coefficients produced by the `bfg` code indicate transience, then this value should increase.

`n` — The number of nodes is given as the second value in the first line of the `loc2glob.dat` file

`n_modes` — User choice, but note that `n_modes` $<$ `nss`

`dyn_avg and dyn_phi` — These can be determined by finding their definitions in the `uns3d.input` file.

`ncpu` — If parallel was set to `.true.` then `ncpu` is the number of processors used to run the `UNS3D POD` code. If that is unknown, see the suffix `_xxx` at the end of the `rhosnaps.dat` files in the `out` folder. The number of processors is equal to the final number plus one.

#### *C.2.2.2 `bfg` Output Files*

The `bfg` code will generate output files given by what was inputted into the command line.

`basis_file###.dat` — Basis function ### for the provided snapshot file

`tc_file###.dat` — Projected time coefficients for the provided snapshot file

`energy_spectrum` — The energy spectrum of the provided snapshot file

The basis function files will contain a `_###` appended to the end of the file name if the parallel flag was set to true. The `###` indicates the processor number. For dynamic functions, each column

will contain the value of the subfunction, where the subfunction is $\tilde{\boldsymbol{\varphi}}_k^j(\mathbf{x})$, $k \in [0, d]$ and $j \in [1, m]$, as shown in (2.10). The zeroth basis function file corresponds to the average.

The time coefficient files contain the time coefficients corresponding to the basis functions generated for the snapshots sampled. Each row corresponds to a point in time.

### C.3 Interpolation for Off-Reference Conditions

This section describes the methodology used to generate basis functions for off-reference conditions and how to use the code `grassman`.

#### C.3.1 `grassman` Code

The code `grassman` has been provided to generate spatial functions for off-reference conditions. The input files for this code are:

1. `loc2glob.dat_nnn`, where `nnn` is the number of processors used in the simulation.

2. `interpolator_input.nml`; the structure of the file is:

## &interpolator_input

| | | |
|---|---|---|
| `n_modes` | I | Number of basis functions |
| `n` | I | Number of nodes |
| `n_conditions` | I | Number of flow conditions known |
| `two_d` | I | Whether flow is two-dimensional |
| `parallel` | L | Whether the basis functions are distributed across multiple processors |
| `dyn_avg` | L | Whether the average is dynamic |
| `dyn_phi` | L | Whether the basis functions are dynamic |
| `n_p` | I | The number of processors if parallel |
| `collectively` | L | Whether to interpolate the basis functions collectively or individually (recommend collectively) |

## &conditions

| | | |
|---|---|---|
| `m0` | R | Condition of interest |
| `m(1)` | R | Known first condition |
| `m(2)` | R | Known second condition |
| ⋮ | ⋮ | ⋮ |
| `m(n_conditions)` | R | Known `n_conditions` condition |

## &directories

| | | |
|---|---|---|
| `loc2glob_dir` | C | Location of `loc2glob.dat` file |
| `basis_dir(1)` | C | Location of basis functions for first condition |
| `basis_dir(2)` | C | Location of basis functions for second condition |
| ⋮ | ⋮ | ⋮ |
| `basis_dir(n_conditions)` | C | Location of basis functions for `n_conditions` condition |

## &basisfiles

| | | |
|---|---|---|
| `basisfile(1)` | C | Name of the density $\rho$ or specific volume $\zeta$ basis file |
| `basisfile(2)` | C | Name of the $x$-velocity $u$ basis file |
| `basisfile(3)` | C | Name of the $y$-velocity $v$ basis file |
| `basisfile(4)` | C | Name of the pressure $p$ basis file |
| `basisfile(5)` | C | Name of the $z$-velocity $w$ basis file |

For the `basisfiles` input, the name of the basis file will match the `basis_file` input for the `bfg` code. It is important that the $z$-velocity basis file be put last such that the `two_d` option can work properly.

3. Basis functions generated by running the `bfg` code on the full-order model snapshots

#### *C.3.1.1 `grass` Example*

A test case has been provided in folder `grassman/dat` that contains two sets of snapshot files.

1. In the `grassman` directory, compile the code by typing `make`. This will yield the executable `grass`.

2. Go to the folder `dat`. This folder consists of three folders: `input`, `output`, and `run`. The `input` folder consists of the folders `1` and `2`: folders `1` and `2` contain the basis functions generated by using the full-order model snapshots. In the `run` folder are the `interpolator_input.nml` and `loc2glob.dat_008` files. The `output` folder contains basis functions produced by the `grass` code.

3. Copy or symbolically link the executable `grass` into the `run` folder.

4. To run the code, type `grass` in the `run` folder. This run generates the interpolated basis functions.

5. To verify the run was successful, the content of the files `r?m00?.dat_00?` that contain the interpolated basis functions can be checked against the files provided in the `output` folder.

### C.4 Starting and Executing `unsromBC`

This chapter goes over the steps to start and execute the `unsromBC` code and finishes with example cases that are provided in the $\zeta$-ROM package.

### C.4.1 Steps to start and execute

1. Run the `UNS3D POD` code to produce the snapshot files. There should be five snapshot files with the following names:

   `rhosnaps.dat` - contains snapshots of $\rho$

   `rhousnaps.dat` - contains snapshots of $\rho u$

   `rhovsnaps.dat` - contains snapshots of $\rho v$

   `rhowsnaps.dat` - contains snapshots of $\rho w$

   `rhoEsnaps.dat` - contains snapshots of $\rho E$

   Even if the case is two-dimensional, a snapshot file of `rhowsnaps.dat` should exist of the same size as the other snapshot files.

2. Link in the `uns3d .mesh` file and connectivity file `c2n.def`

3. If the `uns3d` code was ran in parallel, link in the the `loc2glob` file. Link in the `combineout` executable and copy the executable script `combine` from the `splitcombine` directory. The `combine` executable file may need to be edited depending on the . extension for the `loc2glob` file. To run the executable file `combine` use the following command sequence:

   ```
   ./combine number_of_processors number_of_snapshots
   ```

   The `number_of_processors` input is a three character numerical sequence matching that of the `loc2glob` file.

4. Link in the `convert` executable from the `convert` directory. Copy in the input file `inputPP` from the `convert` directory and modify it as needed for the case. Execute the `convert` code.

   ```
   ./convert inputPP
   ```

5. Create a directory for the POD modes in the same directory as the snapshot files and outputs of the `convert` code. Enter the created directory.

6. *(Only for blade deformation)* Go to the `mesh` folder. Link in the executable `findsurface` from the `findsurface` directory. Execute it for one processor.

```
./findsurface 1
```

   This will create a file `surfacenode.dat`. Link this file into the POD modes directory.

7. Link in all outputs of the `convert` code except for `out.plt`. Link in the `uns3d .mesh` file and connectivity file `c2n.def`. Link in `numsnaps.dat` and `gammas.dat` produced by `uns3d`.

8. Link in the executable for the `bfg` code. Copy in the input file `bfg_input.nml` and modify it for your case. **It is important to note** that the `skip_aux` input is only for the `numsnaps.dat` and `gammas.dat` files, and that `nss` should be the size of your `t.dat` file. Copy in the executable script `run_zeta` from the `bfg` file. Execute the executable script `run_zeta`.

9. Link in the `outlet_ij.def` file from the `mesh` directory. If this file does not exist, skip to Step 10 and run `unsromBC` with `solveODE` and `writeout` set to `false`. Go to Step 9 if desired.

10. *(Optional: to develop the approximate prescribed boundary equations)* Copy in the input for the `BC_interpolation` code, "`BCinterp_input`" and modify as needed for the case. Execute the `BC_interpolation.m` code. This will output the approximate prescribed boundary equations and `FFT.dat` file.

11. Link in the $\zeta$-ROM executable "`unsromBC`" from the `src` directory. Copy in its inputs from the directory `inputfiles`.

12. Modify the files `input` and `const.dat` for your case. To execute `unsromBC` type

```
./unsromBC input
```

in the terminal.

13. The `unsromBC` code will output while running: the time, the first modal time coefficients for each state variable, the error between the prescribed boundary condition and the reconstructed solution $\boldsymbol{K}$, the value of the penalty parameter, and the value of the artificial dissipation parameter for pressure and specific volume respectively.

14. If `writeout` is set to `true`, then `uns3d` will calculate the error of the ROM solution to the FOM solution contained in the converted snapshot files.

### C.4.2 Example Cases

Two examples cases are provided in the `example` folder in the `unsromBC` directory: `2d10thstd` and `3drotor67`. The `2d10thstd` case corresponds to the 10th standard configuration with a blade plunging deformation. The `3drotor67` case corresponds to an outlet pressure oscillation applied to a transonic compressor fan referred to as NASA Rotor 67. The cases will need to be extracted using the `tar` command.

The layout of both cases is similar. Both files contain six common directories:

`eout` - contains the terminal output of the `UNS3D POD` code

`input` - contains the input files of the `UNS3D POD` code

`mesh` - contains all files pertaining to the mesh

`out` - contains all written outputs of the `UNS3D POD` code

`plt` - contains tecplot files written by the `UNS3D POD` code

`txt` - contains text files written by the `UNS3D POD` code

among other auxiliary files. This layout is similar to any case ran using the `UNS3D POD` software.

The user will find that the steps given in Sec. C.4.1 have already been performed in these examples. It is recommended that the user attempt to repeat those steps to better learn how to run the code. If the user wishes to just run the code, follow these steps:

1. Untar the `mesh` directory and out directory

```
tar -xvf mesh.tgz
tar -xvf out.tgz (or out_1.tgz and out_2.tgz)
```

2. Move to the `out` directory

```
cd out
```

3. Untar the `convertoutput.tgz` file

```
tar -xvf convertoutput.tgz
```

4. Move to the `modes` directory

```
cd modes
```

5. The basis functions and time coefficients have all been compressed and will need to be extracted

```
tar -xvf basisfunctions.tgz
tar -xvf timecoefficients.tgz
```

6. Run `unsromBC`

```
./unsromBC input
```

The output files can be compared to those in the `output` directory contained in the `modes` folder.

## C.5 Input and Output Files for the $\zeta$-ROM

This section describes the input and output files for the $\zeta$-ROM .

### C.5.1 $\zeta$-ROM Input Files

#### *C.5.1.1 Main input file*

The main input file for the $\zeta$-Coordinate POD-ROM is written using the Fortran namelist format. Multiple input blocks are used to specify the input parameters. The first column gives the name of the variable. The second gives the variable type: integer (I), real(R), logical (L), or character string (C). The final column gives the variable description, and, in some cases, a default value.

**&card1**

| | | |
|---|---|---|
| `nt` | I | Number of snapshots<br>0 - *default* (automatically detect size of snapshots from `t.dat` file) |
| `n` | I | Number of nodes |
| `nz` | I | Number of modes for $\zeta$ |
| `nu` | I | Number of modes for $u$ |
| `nv` | I | Number of modes for $v$ |
| `nw` | I | Number of modes for $w$ |
| `np` | I | Number of modes for $p$ |
| `et` | R | End time of integration<br>0 - *default* (automatically detect end time from `t.dat` file) |

| | | |
|---|---|---|
| `twod` | L | Flag for whether the case is two-dimensional |
| `wlsqr_pwr` | R | Weighted least squares power |
| `gridfile` | C | Name of input `.mesh` file (see Sec. C.5.1.3) |
| `debug` | L | Flag for debugger mode |
| `wrtcoeff` | L | Flag to write out coefficients |
| `readKder` | L | *Not Recommended* Flag to read in precalculated derivatives of the basis functions from files (see Sec. C.5.1.3) |
| `odecheck` | L | Flag to compare exact $\dot{a}$ with the $\dot{a}$ that is calculated by the $\zeta$-coordinate POD-ROM |
| `solvecheck` | L | Flag to compute the integration of the ODE system |
| `writeout` | L | Flag to write out the results of the integration |
| `movie` | L | Flag to write out a Tecplot movie during integration |
| `deform` | L | Flag for whether a deformation of the entire mesh is occurring (not applicable for deforming blades) |
| `blade` | L | Flag for whether a deformation of a blade is occurring |
| `harmonic` | L | Flag for chord-wise bending |

## &ODEsolve

| | | |
|---|---|---|
| `odesolver` | I | Type of ODE solver<br>0 - ODEPACK<br>1 - Forward Euler<br>-1 - Backward Euler |
| `indst` | I | Index of start time for integration |
| `dt` | R | Time step<br>0 - *default* (automatically detect size of time step from `t.dat` file) |

| | | |
|---|---|---|
| `dtscale` | R | Scalar multiple that scales the time step |
| `itol` | I | *ODEPACK Input* Flag that indicates the type of local error control to be performed.<br>1 - scalar rtol and scalar atol<br>2 - scalar rtol and array atol<br>3 - array rtol and scalar atol<br>4 - array rtol and array atol |
| `rtol` | R | *ODEPACK Input* The local relative error tolerance parameter for the solution |
| `atol` | R | *ODEPACK Input* The local absolute error tolerance parameter for the solution |
| `itask` | I | *ODEPACK Input* An index that specifies the task to be performed |
| `iopt` | I | Integer flag that specifies if any optional input is being used<br>0 - no optional input parameters have been set<br>1 - optional input parameters have been set |
| `mf` | I | Method flag that indicates both the integration method and corrector iteration technique to be used<br>10 - nonstiff solver<br>21 - stiff solver, internally generated Jacobian<br>22 - stiff solver, Jacobian provided in `jac.f90` |
| `mxstep` | I | Maximum number of integration steps allowed on one call to ODEPACK |

| | | |
|---|---|---|
| `typeBC` | I | Type of enforcement of boundary conditions<br>0 - no enforcement of boundary conditions<br>-1 - penalty enforcement of boundary conditions using Eq. (4.7).<br>1 - penalty enforcement of boundary conditions only on $\zeta$ using Eq. (4.8).<br>2 - penalty enforcement of boundary conditions only on $u$ using Eq. (4.8).<br>3 - penalty enforcement of boundary conditions only on $v$ using Eq. (4.8).<br>4 - penalty enforcement of boundary conditions only on $w$ using Eq. (4.8).<br>5 - penalty enforcement of boundary conditions only on $p$ using Eq. (4.8).<br>6 - penalty enforcement of boundary conditions only on $\zeta$ and $p$ using Eq. (4.8).<br>7 - penalty enforcement of boundary conditions on all state variables |
| `tpconst` | R | Adds this value to the penalty parameter for all state variables |
| `scconst` | R | Adds this value to the artificial dissipation parameter for all state variables |

## &flow

| | | |
|---|---|---|
| `pback` | R | Outlet pressure value |
| `POF` | R | Frequency of imposed unsteadiness (unused if `FFT` is set to `.true.`) |

| | | |
|---|---|---|
| `POA` | R | Amplitude of imposed unsteadiness (unused if `FFT` is set to `.true.`) |
| `phase` | R | Phase of imposed unsteadiness (unused if `FFT` is set to `.true.`) |
| `pref` | R | Reference pressure value for nondimensionalization |
| `Tref` | R | Reference temperature value for nondimensionalization |
| `ptot` | R | Stagnation pressure value |
| `Ttot` | R | Stagnation temperature value |
| `lref` | R | Reference length for nondimensionalization |
| `omegax` | R | Wheel speed of the blade rotating about the $x$-axis [rpm] |

## &stability

| | | |
|---|---|---|
| `stablecheck` | L | Flag to check stability of ODE system with current options |
| `stol` | R | The relative error desired in the approximate solution for finding the critical points |
| `jovbl` | C | Flag to calculate left eigenvectors<br>"N" left eigenvectors are not computed<br>"V" left eigenvectors are computed |
| `jobvr` | C | Flag to calculate right eigenvectors<br>"N" right eigenvectors are not computed<br>"V" right eigenvectors are computed |

## &approxbc

| | | |
|---|---|---|
| FFTflag | L | Flag to calculate approximate prescribed boundary conditions using FFT |
| zback | R | Mean specific volume for the prescribed B.C. in Eq. (5.13) |
| zPOA | R | Amplitude of specific volume prescribed B.C. in Eq. (5.13). -1 - use the isentropic approximation for $\zeta$ as the prescribed boundary condition |
| zphase | R | Phase of specific volume prescribed B.C. in Eq. (5.13) |
| uback | R | Mean *x*-velocity for the prescribed B.C. in Eq. (5.13) |
| uPOA | R | Amplitude of *x*-velocity prescribed B.C. in Eq. (5.13) |
| uphase | R | Phase of *x*-velocity prescribed B.C. in Eq. (5.13) |
| vback | R | Mean *y*-velocity for the prescribed B.C. in Eq. (5.13) |
| vPOA | R | Amplitude of *y*-velocity prescribed B.C. in Eq. (5.13) |
| vphase | R | Phase of *y*-velocity prescribed B.C. in Eq. (5.13) |
| wback | R | Mean *z*-velocity for the prescribed B.C. in Eq. (5.13) |
| wPOA | R | Amplitude of *z*-velocity prescribed B.C. in Eq. (5.13) |
| wphase | R | Phase of *z*-velocity prescribed B.C. in Eq. (5.13) |

## &dynamic

| | | |
|---|---|---|
| np_var | I | Vector of length $D + 2$ containing the number of parameters per state variable |
| dyn_avg | I | Flag vector of length $D + 2$ for whether each state variable is approximated using dynamic average |
| dyn_phi | I | Flag vector of length $D + 2$ for whether each state variable is approximated using dynamic basis |

### *C.5.1.2 Secondary input file: "const.dat"*

This file contains the values for the artificial dissipation parameters and how many modified number of modes each variable uses in its ODE. The format is as follows.

```
# Dissipation Constants
sc(1)
sc(2)
sc(3)
sc(4)
sc(5)
# ODE Modes
nz Mnu Mnv Mnw Mnp
Xnz nu Xnv Xnw Xnp
Ynz Ynu nv Ynw Ynp
Znz Znu Znv nw Znp
Enz Enu Env Enw np
```

In this file, `sc(1)` to `sc(5)` are real values. If the user wants to calculate the dissipation parameter, $\tilde{\nu}$, using Eq. (4.11), then set each value of `sc` to -1. In `ODE Modes`, the rows correspond to each state variable's ODE: conservation of mass, conservation of momentum, and conservation of energy. Each column corresponds to the number of modified number of modes for each state variable. The diagonal consisting of `nz`, `nu`, `nv`, `nw`, and `np` correspond to $n^{\zeta}$, $n^{u}$, $n^{v}$, $n^{w}$, and $n^{p}$ respectively. Table C.12 shows an example with marked columns and rows. In this case, $n^{\zeta} = 1, n^{u} = 3, n^{v} = 2, n^{w} = 2$, and $n^{p} = 1$. There are a modified number of modes for $v$ in Mass and Energy where `Mnv`=1 and `Env`=1. Recall that the modified number of modes cannot be greater than what is given by $n^{q_k}, k = [1, D + 2]$.

| | $\zeta$ | $u$ | $v$ | $w$ | $p$ |
|---|---|---|---|---|---|
| Mass | 1 | 3 | 1 | 2 | 1 |
| $x$-Momentum | 1 | 3 | 2 | 2 | 1 |
| $y$-Momentum | 1 | 3 | 2 | 2 | 1 |
| $z$-Momentum | 1 | 3 | 2 | 2 | 1 |
| Energy | 1 | 3 | 1 | 2 | 1 |

Table C.12: Example of `ODE Modes` in `const.dat` file.

#### *C.5.1.3 Auxiliary input files*

This section provides a list of files that the $\zeta$-ROM code reads in apart from the `input` and `const.dat` files. All files are either from `uns3d`, the `convert` code, or from the `BC_interpolation` code.

`z.dat` Snapshot file of specific volume produced by the FOM. Used for calculating the error between the ROM and the FOM

`u.dat` Snapshot file of $x$-velocity produced by the FOM.

`v.dat` Snapshot file of $y$-velocity produced by the FOM.

`w.dat` Snapshot file of $z$-velocity produced by the FOM.

`p.dat` Snapshot file of pressure produced by the FOM.

`grid.dat` Grid file produced by the `convert` code

`t.dat` Nondimensional time file produced by the `convert` code

`vol.mesh` The `uns3d` grid file

`c2n.def` The `uns3d` node connectivity file

`bladenodes.dat` File containing the blade nodes from which the approximate prescribed boundary conditions were developed. Produced by the `BC_interpolation` code

`FFT.dat` File containing the FFT coefficients found by the `BC_interpolation` code

### C.5.2 $\zeta$-Coordinate POD-ROM Output Files

The following files contain the time and the **exact** time coefficients, $a^{q_i}_{\text{exact}}$, for each variable of the state vector.

- `TCtz.dat`: time, $a^{\zeta_1}_{\text{exact}}, \ldots, a^{\zeta_{n^\zeta}}_{\text{exact}}$
- `TCtu.dat`: time, $a^{u_1}_{\text{exact}}, \ldots, a^{u_{n^u}}_{\text{exact}}$
- `TCtv.dat`: time, $a^{v_1}_{\text{exact}}, \ldots, a^{u_{n^v}}_{\text{exact}}$
- `TCtw.dat`: time, $a^{w_1}_{\text{exact}}, \ldots, a^{u_{n^w}}_{\text{exact}}$
- `TCtp.dat`: time, $a^{p_1}_{\text{exact}}, \ldots, a^{u_{n^p}}_{\text{exact}}$

The following files contain the time and the **calculated** time coefficients, $a^{q_i}$, for each variable of the state vector.

- `TCtz_out.dat`: time, $a^{\zeta_1}, \ldots, a^{\zeta_{n^\zeta}}$
- `TCtu_out.dat`: time, $a^{u_1}, \ldots, a^{u_{n^u}}$
- `TCtv_out.dat`: time, $a^{v_1}, \ldots, a^{v_{n^v}}$
- `TCtw_out.dat`: time, $a^{w_1}, \ldots, a^{w_{n^w}}$
- `TCtp_out.dat`: time, $a^{p_1}, \ldots, a^{p_{n^p}}$

The following files contain the spatial derivative of the basis functions of each variable of the state vector. The columns are in the following order: all modes of $\partial\phi/\partial x$, all modes of $\partial\phi/\partial y$, all modes of $\partial\phi/\partial z$, all modes of $\partial^2\phi/\partial x^2$, all modes of $\partial^2\phi/\partial y^2$, and all modes of $\partial^2\phi/\partial z^2$.

- `dKLz.dat` Spatial derivatives of $\zeta$ modes

- `dKLu.dat` Spatial derivatives of $u$ modes
- `dKLv.dat` Spatial derivatives of $v$ modes
- `dKLw.dat` Spatial derivatives of $w$ modes
- `dKLp.dat` Spatial derivatives of $p$ modes
- `err.plt` Tecplot file showing the time-averaged error of the $\zeta$-ROM vs. the FOM
- `max.plt` Tecplot file showing the maximum error of the $\zeta$-ROM vs. the FOM at each nodal point
- `outmovie.plt` Tecplot movie file showing the progress of the $\zeta$-ROM as it integrates the ODE system
- `KLmodes_#.plt` Tecplot file showing the basis functions of the # mode
- `youtput.dat` File containing the last calculated time coefficients listed in rows by order: all modes of specific volume, all modes of *x*-velocity, all modes of *y*-velocity, all modes of *z*-velocity, and all modes of pressure
- `time.dat` File containing the dimensional time
- `tauvst.dat` File outputting the penalty parameter of pressure versus time
- `Kcal.dat` File outputting the calculated error for pressure at the next time step used in the penalty method in Eq. (4.8). Column 1 is time, columns 2 to np+1 is the predicted error at the next time step for each mode, column np+2 is the reconstructed pressure, and column np+3 is the prescribed boundary condition for pressure
- `Kact.dat` File outputting the actual error for pressure at the current time step for comparing to the values outputted in `Kcal.dat`. Column 1 is the number

of modes for pressure, column 2 is time, columns 3 to np+2 are the predicted errors for each mode, column np+3 is reconstructed pressure, and column np+4 is the prescribed boundary condition for pressure

- `ytoutput#.dat` File containing the calculated time coefficients along with time. The first column contains dimensionless time. The second column and onward are in the order: all modes of specific volume, all modes of $x$-velocity, all modes of $y$-velocity, all modes of $z$-velocity, and all modes of pressure

- `maxerr.dat` File containing the maximum error at each time step for each state variable along with the nodal location it occurred in the mesh. Column 1 is the dimensionless time, columns 2-6 are the maximum error of specific volume, $x$-velocity, $y$-velocity, $z$-velocity, and pressure respectively, and columns 7-12 are the locations of the maximum error for specific volume, $x$-velocity, $y$-velocity, $z$-velocity, and pressure respectively

### C.6 Input and Output Files for `convert`

#### C.6.1 `convert` Input File

**&card1**

| | | |
|---|---|---|
| `gridfile` | C | `uns3d.mesh` file |
| `nruns` | I | Number of snapshots |
| `ntrans` | I | Number of transients in the snapshots (to skip) |
| `zlayer` | I | *Not Recommended to use unless the* `.mesh` *file is changed* (For 2D) If ran 3D, but want a 2D solution, select which slice in the $z$-direction to use |
| `L` | R | *Not Recommended to use unless the* `.mesh` *file is changed* (For 2D) Length of the domain (to detect when each $z$-layer ends) |

| | | |
|---|---|---|
| `movie` | L | Flag to output a Tecplot movie of the snapshots |
| `R` | R | Universal gas constant in SI units |
| `Tref` | R | Reference static temperature |
| `Pref` | R | Reference static pressure |
| `lref` | R | Reference length |
| `omegax` | R | Wheel speed of the blade rotating about the $x$-axis |

#### *C.6.1.1 Auxiliary Files*

- `numsnaps.dat` File containing the time for each snapshot
- `rhosnaps.dat` File containing the snapshots of $\rho$. Each snapshot is outputted at its respective time step for all nodes and appended onto the previous snapshot.
- `rhousnaps.dat` '...' $\rho u$ '...'
- `rhovsnaps.dat` '...' $\rho v$ '...'
- `rhowsnaps.dat` '...' $\rho w$ '...'
- `rhoEsnaps.dat` '...' $\rho E$ '...'
- `c2n.def` The `uns3d` nodal connectivity file

### C.6.2 `convert` Output File

- `z.dat` File containing the snapshots of **nondimensionalized** $\zeta$. Each snapshot is outputted at its respective time step for all nodes and appended onto the previous snapshot.
- `rho.dat` '...' $\rho$ '...'
- `u.dat` '...' $u$ '...'

- `v.dat` '...' $v$ '...'
- `w.dat` '...' $w$ '...'
- `p.dat` '...' $p$ '...'
- `t.dat` File containing the **nondimensional** time for each snapshot
- `grid.dat` Grid file containing the $x$, $y$, and $z$-coordinates for each nodal point
- `out.plt` Tecplot file outputting the FOM solution at the last snapshot

### C.7 `splitout` and `combineout`

The `UNS3D POD` code writes its outputs based off of processor number when run in parallel. Sometimes the user may wish to combine these files into one. Or, the user may wish to split a single file into multiple files based off of processor number. For these cases, the `split_out` and `combineout` codes can be used.

Both codes are contained in the `split_out` directory. The codes are compiled using gfortran. If another compiler is desired, the user will need to modify the makefile within the directory. To compile the codes type

```
make
```

in the directory. The makefile is designed to compile both codes.

#### C.7.1 `splitout`

The `splitout` code takes a single file and splits it to be read by multiple processors based off of a `loc2glob` file. When executing the `splitout` code without any command line input arguments, the `splitout` code will output the following: This lets the user know how to run the code. There are three inputs: `filename`, `filetype`, and `l2gfile`.

- `filename` - The name of the file to be split (without the . extension)

**Program 1** `splitout` line argument output

```
 ------------------------------------------------------------
 |  The correct usage of this executable is as follows:     |
 |     splitout <filename> <filetype> <l2gfile>,            |
 |                                                          |
 |  <filename> -- File to be split                          |
 |  <filetype> -- Can be "out", "def", "tur", or "#",       |
 |            where "#" refers to the number of values/node |
 |  <l2gfile>  -- Output file from splitmesh                |
 ------------------------------------------------------------
```

- `filetype` - The type of file: "out", "def", "tur", or "#", where "#" refers to the number of values/node. This code does not work on snapshot files. The `filetype` refers to the . extension of the `filename`.

- `l2gfile` - The `loc2glob` file read in by the `UNS3D POD` code.

### C.7.2 `combineout`

The `combineout` code takes multiple files split by processor number and combines them into a single file using a provided `loc2glob` file. When executing the `combineout` code without any command line input arguments, the `combineout` code will output the following: This lets

**Program 2** `combineout` line argument output

```
 ------------------------------------------------------------
 |  The correct usage of this executable is as follows:     |
 |   combineout <filename> <filetype> <l2gfile> [<c2nfile>],|
 |                                                          |
 |  <filetype> -- Can be "out", "def", "tur", "plt",        |
 |            "snaps", or "#", where "#" refers to the      |
 |            number of values/node                         |
 |  <filename> -- Base name of files to be combined         |
 |  <l2gfile>  -- Output file from splitmesh                |
 |  <c2nfile>  -- Name of c2n connectivity (plt only)       |
 |  <niter>    -- Number of iterations (snaps only)         |
 ------------------------------------------------------------
```

the user know how to run the code. There are three to five inputs depending on the `filetype`: `filename`, `filetype`, `l2gfile`, `c2nfile`, and `niter`.

- `filename` - The name of the file to be split (without the . extension)
- `filetype` - The type of file: "out", "def", "tur", "plt", "snaps", or "#", where "#" refers to the number of values/node. For all except "snaps", the `filetype` refers to the . extension of the `filename`. The "snaps" input is for combining snapshot files.
- `l2gfile` - The `loc2glob` file read in by the `UNS3D POD` code.
- `c2nfile` - The connectivity file `c2n.dex` read in by the `UNS3D POD` code.
- `niter` - The number of snapshots.