8-2011

The Biglobal Instability of the Bidirectional Vortex

Joshua Will Batterson
jbatters@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Joshua Will Batterson entitled "The Biglobal Instability of the Bidirectional Vortex." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Aerospace Engineering.

Joseph Majdalani, Major Professor

We have read this dissertation and recommend its acceptance:

Trevor Moeller, Roy Schulz, Christian Parigger

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

The Biglobal Instability of the Bidirectional Vortex

A Dissertation

Presented for the

Doctor of Philosophy Degree

The University of Tennessee, Knoxville

Joshua Will Batterson

August 2011

I know well who shaped my path. This is yours as much as it is mine.

Acknowledgements

First, I would like to acknowledge the University of Tennessee Space Institute for giving me a very unique opportunity to pursue my academic goals. It is an unparalleled environment for nurturing challenging and ambitious research and education. More specifically, special recognition is held for Dr. Majdalani who accompanied me on this grueling path as my advisor and teacher. Dr. Moeller, Dr. Parigger, and Dr. Schulz were gracious enough to serve on my committee and lend their expertise to bring my efforts to fruition. Dr. Kupershmidt, Dr. Flandro, and others built the foundation for me to pursue and excel in truly advanced research. Several students, including Dr. Eric Jacob, Dr. Tony Saad, and Nadim Zgheib supported me with mathematical and scientific insight along the way. I must also recognize the National Science Foundation for funding me briefly during my pursuits, and UTSI for providing the majority of my funding. Without the cooperative relationship with Dr. Casalis and Dr. Chedevergne at ONERA this research avenue may have never been possible. Lastly, where would I be without the support and patience of my family. The anxiousness I have felt for my graduation is only exceeded by theirs!

To be free is to be simple. I am not simple...

Abstract

The paradigm shift toward two-dimensional techniques with the ability to accommodate fully three-dimensional base flows is a necessary step toward modeling complex, multidimensional flowfields in modern propulsive applications. Here, we employ a two-dimensional spatial waveform with sinusoidal temporal dependence to reduce the three-dimensional linearized Navier-Stokes equations to their *biglobal* form. Addressing hydrodynamic stability in this way circumvents the restrictive parallel-flow assumption and admits boundary conditions in the streamwise direction. Furthermore, the following work employs a full momentum formulation, rather than the conventional streamfunction form, thus accounting for a nonzero tangential mean flow velocity.

Specifically, three bidirectional vortex models, stemming from two unique families, are analyzed for stability. The complex-lamellar model exhibits linear axial and sinusoidal radial dependence. The two Beltramian type solutions, one axially linear and one harmonic, have radial dependence of the Bessel type. The three tangential velocity components are arrived at by matching viscous boundary layers to previously inviscid outer solutions. While all three match experimental data, the two Beltramian models display remarkable agreement.

Parametric studies are conducted for two values of the inlet parameter, κ. For smaller κ (greater swirl intensity), an appreciable stabilizing effect is seen in the spectral predictions. Hydrodynamic frequencies tend to collect nearer the critical line and are lower frequency than those predicted for larger κ. Significant differences are also found in the unstable waveforms.

When swirl is less dominant, the hydrodynamic wave oscillates about the streamlines of the base flow. The largest oscillations appear near the headwall and are drawn into the core vortex. Oscillation amplitudes diminish in the streamwise direction, suggesting spatial stability. These results disappear as swirl becomes more dominant. The effect of the axial and radial base flow velocities is apparently negated by a nearly axially invariant tangential velocity. In this case, when oscillations appear, they are smaller in amplitude and are confined to regions of increased vorticity (i.e. the sidewall or downstream). In general, we conclude that the overall stability characteristics can be substantially increased by minimizing the inlet parameter and maximizing the swirl intensity.

Contents

1 Introduction **1**

1.1 The Bidirectional Vortex 1

1.2 Instability 4

1.2.1 On the Vortico-Acoustic Wave Formulation 7

1.2.2 On the Hydrodynamic Wave Formulation 8

1.2.3 On Linear Hydrodynamic Stability Theory 14

1.2.4 Parallel Flow Effects 20

2 Vortex Flows **24**

2.1 An Introduction to Vortex Flows 25

2.2 The Complex Lamellar Bidirectional Vortex 29

2.2.1 The Invsicid Solution 30

2.2.2 Viscous Corrections 35

2.3 The Beltramian Bidirectional Vortex 44

2.3.1 The Linear Beltramian Solution 48

2.3.2 Viscous Corrections 50

2.3.3 The Harmonic Beltramian Solution 55

2.4 A Comment on the Existence of Multiple Mantles 60

2.4.1 Experimental Validation 64

3 Spectral Collocation Methods **66**

3.1 Polynomial Approximation and Interpolation 67

3.2 Chebyshev Polynomials 68

3.3 Pseudo-Spectral Derivatives 72

3.4 Solving ODEs with Chebyshev Collocation 78

3.4.1 Example: A First Order ODE with Variable Coefficients 82

3.4.2 Example: A Second Order BVP with Variable Coefficients 84

3.4.3 Example: A System of Differential Equations 87

3.5 Eigenvalue Problems with ODEs 92

3.5.1 Example: Eigenvalues of the Bessel Equation 93

3.6 Solving PDEs with Chebyshev Collocation 98

3.6.1 Example: A Parabolic PDE with Variable Coefficients 108

3.6.2 Example: The 2D Poisson Equation 113

3.6.3 Example: A System of PDEs 114

3.7 Eigenvalue Problems with PDEs 119

3.7.1 Example: The Helmholtz equation 120

3.8 Closing Remarks on Spectral Methods 122

4 Eigensolvers **128**

4.1 Calculating Eigenvalues 129

4.2 Matrix Preconditioning 130

4.2.1 Balancing a Single Matrix 132

4.2.2 Balancing the Generalized Eigenvalue Problem 132

4.2.3 Segregating Nonzero Elements 134

4.3 Matrix Reductions 135

4.3.1 Reduction of a Single Matrix 136

4.3.2 Reduction of a Matrix Pencil 138

4.3.3 Block Decomposition 139
4.4 Single Matrix Eigensolvers 141
4.4.1 The Power Method 141
4.4.2 The Inverse Power Method 142
4.4.3 The QR and LR Methods 143
4.5 Generalized Eigensolvers 145
4.5.1 Example: Implementation of an LZ Eigensolver 149
4.6 Closing Remarks on Eigensolvers 156

5 Local Nonparallel Stability Analysis of the Bidirectional Vortex 159
5.1 Deriving the Spectral LNP Equations 160
5.2 Code Validation and Grid Refinement 165
5.3 The Complex-Lamellar Bidirectional Vortex 167
5.3.1 Axisymmetric Spectrum 168
5.3.2 Asymmetric Spectrum 169
5.3.3 Multiple Mantles 171
5.3.4 Amplified Frequencies 173
5.4 The Linear Beltramian Bidirectional Vortex 180
5.4.1 Axisymmetric Spectrum 181
5.4.2 Asymmetric Spectrum 182
5.4.3 Multiple Mantles 182
5.4.4 Amplified Frequencies 184
5.5 The Harmonic Beltramian Bidirectional Vortex 190
5.5.1 Axisymmetric Spectrum 191
5.5.2 Asymmetric Spectrum 191
5.5.3 Multiple Mantles 192
5.5.4 Amplified Frequencies 192

5.6 Closing Remarks on the LNP Applied to the BV 199

6 Biglobal Stability Analysis of the Bidirectional Vortex 203

6.1 Deriving the Spectral Biglobal Equations 204

6.2 On the Hardware Requirements 213

6.3 The Complex-Lamellar Bidirectional Vortex 217

6.3.1 Axisymmetric Spectrum 217

6.3.2 Asymmetric Spectrum 219

6.3.3 Evolution with Time 229

6.4 The Linear Beltramian Bidirectional Vortex 231

6.4.1 Axisymmetric Spectrum 231

6.4.2 Asymmetric Spectrum 232

6.4.3 Evolution with Time 240

6.5 The Harmonic Beltramian Bidirectional Vortex 242

6.5.1 Axisymmetric Spectrum 242

6.5.2 Asymmetric Spectrum 243

6.5.3 Evolution with Time 248

6.6 Closing Remarks on the BG Approach 249

6.6.1 Considering the Waveforms 249

6.6.2 Data Reduction 253

6.6.3 On Chamber Length 255

6.6.4 Quantifying Numerical Error 256

7 Conclusions 258

7.1 Comparing the LNP and Biglobal Solutions 259

7.1.1 The Effect of the Tangential Velocity 262

7.2 Unstable Frequencies 266

7.3 Eigensolver Implementation 268
7.4 Future Work 268

Bibliography **271**

A Derivations **286**
A.1 Deriving the 1-D Cylindrical LNP Equations 287
A.2 Deriving the Cylindrical Biglobal Stability Equations 289

B Numerical Codes **293**
B.1 Polynomial Interpolation 294
B.2 Chebyshev Interpolating Polynomial Generator 295
B.3 Chebyshev Pseudo-Spectral Differentiation Matrix Generator 296
B.4 Chebyshev Interpolation 298
B.5 Computing a Spectral Derivative 299
B.6 A First order ODE with Chebyshev Collocation 300
B.7 A Second Order BVP with Chebyshev Collocation 301
B.8 Systems of ODEs with Chebyshev Collocation 303
B.9 Eigenvalue Problems for ODEs 305
B.10 A Parabolic Partial Differential Equation with Variable Coefficients 308
B.11 The Time-Independent Poisson Equation with a Sinusoidal Forcing Function 312
B.12 Systems of PDEs with Chebyshev Collocation 315
B.13 Eigenvalue Problems for PDEs 321
B.14 Single Matrix Balancing 324
B.15 Matrix Pencil Balancing 327
B.16 Segregating Nonzero Elements 331
B.17 Real Symmetric Matrix to Tridiagonal Form 333
B.18 Real Nonsymmetric Matrix to Upper Hessenberg Form 336

B.19 Real/Complex Nonsymmetric Matrix to Upper Hessenberg Form 339
B.20 Matrix Pencil Reduction 342
B.21 Block Decomposition 345
B.22 The Power Method 346
B.23 The Inverse Power Method 349
B.24 The QR Method 352
B.25 The LZ Method 361
B.26 The Generalized LZ Algorithm 376

C Unstable Frequencies **384**
C.1 The Complex-Lamellar Bidirectional Vortex 386
C.1.1 Axisymmetric Spectrum 386
C.1.2 Asymmetric Spectrum 387
C.2 The Linear Beltramian Bidirectional Vortex 390
C.2.1 Axisymmetric Spectrum 390
C.2.2 Asymmetric Spectrum 391
C.3 The Harmonic Beltramian Bidirectional Vortex 394
C.3.1 Axisymmetric Spectrum 394
C.3.2 Asymmetric Spectrum 395

Vita **398**

List of Tables

2.1 Eigenvalues and corresponding mantle locations for even flow reversal mode numbers and an odd number of internal mantles. 61

2.2 Comparison of experimental and computational mantle locations with the complex-lamellar (CL) and Beltramian (BEL) models. 65

3.1 Example collocation points, ξ_i for increasing polynomial order, N. 71

3.2 Spectral roots of $J_0(\mu L)$ for $0 \leq x \leq L$ compared to their accepted values. % $E = |\lambda_e - \mu|/\lambda_e$ where λ_e is the root of $J_0(\lambda_e)$ for $0 \leq x \leq 1$. 97

3.3 The normalized eigenvalues of the Helmholtz equation for $0 \leq x \leq L$ compared to their exact values. % $E = |\lambda_e - \mu|/\lambda_e$ where λ_e is the exact value. 123

3.4 Applicable error values for all examples. 127

4.1 Eigenvalues computed for the given example compared to those computed via Lapack. 156

5.1 The first amplified eigenvalues for $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$. . 180

5.2 The first amplified eigenvalues for $q = 1$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$. 190

5.3 The first amplified eigenvalues for $q = 1$, $\alpha = 0.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. 199

5.4 Backsubstitution and boundary condition error values for all plotted eigenvector examples. 202

6.1 The minimum discretization number, N, to characterize the base flow boundary layer with at least three points and estimated runtime for the QZ eigensolver. All times estimated for an Intel Core2 CPU Quad 6600 @ 2.4GHz with 4 Gb RAM and Win 7x64. 215

6.2 Backsubstitution and boundary condition error values for plotted asymmetric eigenvector examples. 257

C.1 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the axisymmetric complex-lamellar parametric study in Fig. 6.3 for $\kappa = 0.1$. 386

C.2 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for variation in κ (Fig. 6.4). 387

C.3 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for variation in Re (Fig. 6.5). 388

C.4 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for variation in l (Fig. 6.9). 388

C.5 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for multidirectional flow number, m (Fig. 6.11). 389

C.6 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the axisymmetric linear Beltramian parametric study in Fig. 6.14 for $\kappa = 0.1$. 390

C.7 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for variation in κ (Fig. 6.15). 391
C.8 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for variation in Re (Fig. 6.16). 392
C.9 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for variation in l (Fig. 6.20). 392
C.10 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for multidirectional flow number, m (Fig. 6.21). 393
C.11 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the axisymmetric harmonic Beltramian parametric study in Fig. 6.25 for $\kappa = 0.1$. 394
C.12 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for variation in κ (Fig. 6.26). 395
C.13 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for variation in Re (Fig. 6.27). 396
C.14 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for variation in l (Fig. 6.31). 396
C.15 The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for multidirectional flow number, m (Fig. 6.32). 397

List of Figures

1.1 VCCWC schematic. 3

1.2 The three identified types of vortex shedding. 5

1.3 A qualitative comparison of the nonparallel effects of several base flows for $r = 0.5$. 23

2.1 Normalized swirl velocity for several historic vortex models. 26

2.2 The bidirectional vortex geometry showing the coordinate system, characteristic dimensions, and general streamline profile. 31

2.3 The inviscid complex-lamellar bidirectional vortex with $\kappa = 0.103$ and $z = 0.3$. 36

2.4 Streamlines of the complex-lamellar BV. 36

2.5 The viscous complex-lamellar bidirectional vortex with $\kappa = 0.103$ and $z = 0.3$. 45

2.6 The inviscid linear Beltramian bidirectional vortex with $\kappa = 0.103$ and $z = 0.3$. 49

2.7 Streamlines of the linear Beltramian BV. 50

2.8 The viscous linear Beltramian bidirectional vortex with $\kappa = 0.103$ and $z = 1$. 55

2.9 The inviscid harmonic Beltramian bidirectional vortex with $\kappa = 0.103$, $l = 2$, and $z = 0.3$. 57

2.10 Streamlines of the harmonic Beltramian BV. 58

2.11 The viscous harmonic Beltramian bidirectional vortex with $\kappa = 0.103$, $l = 2$, and $z = 1$. 60

2.12 Vector plots of a) linear Beltramian, b) harmonic Beltramian, and c) complex-lamellar models. Here $l = 2$. 63

3.1 Two examples of interpolating polynomials. 67

3.2 Comparison between interpolation with equally spaced (left) and Chebyshev grids (right). The Runge phenomenon can be easily seen. 69

3.3 Chebyshev polynomials and corresponding weight functions for $N = 5$. . . . 72

3.4 Spectral differentiation of $\sin(x)$ for three polynomial orders plotted against the exact derivative. 78

3.5 Spectral solution of Eq. (3.31) for three polynomial orders plotted against the exact solution. 84

3.6 Spectral solution of Eq. (3.34) for three polynomial orders plotted against the exact solution. 87

3.7 The solution to the system of ODEs given in Eq. (3.38) for three collocation numbers. 92

3.8 Unforced and sinusoidally forced damped oscillator for $\omega = 2$, $\Omega = 1$, and $c = \frac{1}{2}$ (3.8a) and corresponding FFT showing the damped natural frequency $\sqrt{\omega^2 - c^2}/2\pi \approx 0.308$ and the forcing frequency $\Omega/2\pi \approx 0.159$ (3.8b). 95

3.9 The eigenvectors plotted against the Bessel function of the first kind for the first five eigenvalues for $N = 15$ over the domain $0 \leq x \leq 10$. 96

3.10 The location of nonzero elements in the product tensor matrix. 101

3.11 The solution to the given parabolic PDE under the boundary conditions given above for $N = 20$. 112

3.12 The solution to the time-independent Poisson equation with a sinusoidal forcing function under the boundary conditions given above for $N = 20$. . . . 114

3.13 The solution to the system of PDEs under the boundary conditions given above for $N = 20$. 120

3.14 The eigenvectors and their corresponding eigenvalues for the Helmholtz equation for $N = 20$. Figures 3.14b–3.14c show the eigenvectors for the double eigenvalue $\mu = 5\pi^2/L^2$. 124

4.1 Eigensolver Flowchart 129

4.2 Eigensolver Flowchart 149

5.1 Grid refinement for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 166

5.2 Comparison of the spectrum for the inviscid versus the viscous solution with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 167

5.3 Axisymmetric parametric study for several input parameters. Here, $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 169

5.4 The effect of higher tangential mode numbers with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 170

5.5 Asymmetric parametric study for several input parameters. Here $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 171

5.6 Illustrating the nearly linear axial dependence of the spectrum for the undamped eigenvalue near $\omega = 35 + 4i$ with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 172

5.7 The effect of multiple mantles on the temporal spectrum for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 172

5.8 Axisymmetric eigenvectors for the first undamped frequency, $\omega = 0.1911 + 0.0625i$, with the input parameters $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values. 173

5.9 Axisymmetric contour plots, with $q = 0$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$. . 174

5.10 Asymmetric eigenvectors for the first undamped frequency, $\omega = 1.6590 + 0.0949i$, with the input parameters $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values. 175

5.11 Asymmetric contour plots, with $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$. . . 176
5.12 Temporal evolution of the first unstable eigenvalue on the axial velocity for the complex-lamellar bidirectional vortex with $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 179
5.13 Axisymmetric parametric study for several input parameters. Here, $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 181
5.14 The effect of higher tangential mode numbers with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 182
5.15 Asymmetric parametric study for several input parameters. Here $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 183
5.16 The effect of multiple mantles on the temporal spectrum for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 184
5.17 Axisymmetric eigenvectors for the first undamped frequency, $\omega = 0.5645 + 0.2122i$, with the input parameters $q = 0$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values. 185
5.18 Axisymmetric contour plots, with $q = 0$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$. . 186
5.19 Asymmetric eigenvectors for the first undamped frequency, $\omega = 0.6931 + 0.1220i$, with the input parameters $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values. 187
5.20 Asymmetric contour plots, with $q = 1$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$. . 188
5.21 Temporal evolution of the first unstable eigenvalue on the axial velocity for the linear Beltramian bidirectional vortex with $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 189
5.22 Axisymmetric parametric study for several input parameters. Here, $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$ unless varied on the graph. . 191
5.23 The effect of higher tangential mode numbers with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$ unless varied on the graph. 192

5.24 Asymmetric parametric study for several input parameters. Here $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$ unless varied on the graph. . 193
5.25 The effect of multiple mantles on the temporal spectrum for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. 194
5.26 Axisymmetric eigenvectors for the first undamped frequency, $\omega = 0.1891 + 0.1000i$, with the input parameters $q = 0$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. See Table 5.4 for error values. 194
5.27 Axisymmetric contour plots, with $q = 0$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$. 195
5.28 Asymmetric eigenvectors for the first undamped frequency, $\omega = 0.6262 + 0.0331i$, with the input parameters $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. See Table 5.4 for error values. 196
5.29 Asymmetric contour plots, with $q = 1$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$. . 197
5.30 Temporal evolution of the first unstable eigenvalue on the axial velocity for the Harmonic Beltramian bidirectional vortex with $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. 198

6.1 Qualitative grid refinement using the linear Beltramian model with $l = 2$, $Re = 10000$, $\kappa = 0.1$ and $q = 1$. 216
6.2 Comparison of the spectrum for the inviscid versus the viscous complex-lamellar solution with $q = 1$, $l = 2$, and $\kappa = 0.1$. 216
6.3 Axisymmetric parametric study for several input parameters. Here $q = 0$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 218
6.4 Asymmetric parametric study for several values of κ. Here $q = 1$, $l = 2$, and $Re = 10,000$. 220
6.5 Asymmetric parametric study for several input parameters. Here $q = 1$, $l = 2$, and $Re = 10,000$. 221
6.6 Spectral results for higher mode numbers. Here $l = 2$ and $Re = 10,000$. . . . 221

6.7 Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.2178 + 0.2940i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. See Table 6.2 for error values. 222
6.8 Asymmetric eigensolutions for the unstable eigenvalue, $0.3443 + 0.4257i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.01$. 223
6.9 Asymmetric eigenvalues for several aspect ratios. Here $q = 1$ and $Re = 10,000$. 225
6.10 Axial waveform of the first unstable eigenvalue of the complex-lamellar bidirectional vortex for different aspect ratios with $q = 1$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$. 226
6.11 Asymmetric parametric study for multi-directional flow. Here $q = 1$, $l = 2$, and $Re = 10,000$. 227
6.12 Axial waveform of the first unstable eigenvalue of the complex-lamellar bidirectional vortex for multidirectional flow with $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. 228
6.13 Temporal evolutions of the instantaneous axial velocity for the first unstable eigenvalue, $\omega = 0.2178 + 0.2940i$, and eigensolution of the complex-lamellar bidirectional vortex with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. 230
6.14 Axisymmetric parametric study for several input parameters. Here $q = 0$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 231
6.15 Asymmetric variation with κ. Here $q = 1$, $l = 2$, $Re = 10,000$. 233
6.16 Asymmetric variation with Reynolds number. Here $q = 1$, $l = 2$, and $Re = 10,000$. 233
6.17 Spectral results for higher mode numbers. Here $l = 2$ and $Re = 10,000$. . . . 234
6.18 Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.2312 + 0.1096i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. See Table 6.2 for error values. 236
6.19 Asymmetric eigensolutions for the unstable eigenvalue, $0.8992 + 0.1617i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.01$. 237

6.20 Spectral results for several values of the aspect ratio. Here $q = 1$ and $Re = 10,000$. 238
6.21 Spectral results for several values multidirectional flow. Here $q = 1$, $l = 2$, and $Re = 10,000$. 238
6.22 Axial waveform of the first unstable eigenvalue of the linear Beltramian bidirectional vortex for different aspect ratios with $q = 1$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$. 239
6.23 Axial waveform of the first unstable eigenvalue of the linear Beltramian bidirectional vortex for multidirectional flow with $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. 240
6.24 Temporal evolutions of the instantaneous axial velocity for the first unstable eigenvalue, $\omega = 0.2312 + 0.1096i$, and eigensolution of the linear Beltramian bidirectional vortex with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. 241
6.25 Axisymmetric parametric study for several input parameters. Here $q = 0$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph. 242
6.26 Asymmetric variation with κ. Here $q = 1$, $l = 2$, and $Re = 10,000$. 243
6.27 Asymmetric variation with Reynolds number. Here $q = 1$, $l = 2$, and $Re = 10,000$. 244
6.28 Spectral results for higher mode numbers. Here $l = 2$ and $Re = 10,000$. . . . 244
6.29 Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.0726 + 0.0549i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. See Table 6.2 for error values. 245
6.30 Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.1734 + 0.1602i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.01$. 246
6.31 Spectral results for several values of the aspect ratio. Here $q = 1$ and $Re = 10,000$. 247
6.32 Spectral results for multidirectional flow. Here $q = 1$ and $Re = 10,000$. . . . 248

6.33 Axial waveform of the first unstable eigenvalue of the harmonic Beltramian bidirectional vortex for different aspect ratios with $q = 1$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$. 249
6.34 Axial waveform of the first unstable eigenvalue of the harmonic Beltramian bidirectional vortex for multidirectional flow with $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. 250
6.35 Temporal evolutions of the instantaneous axial velocity for the first unstable eigenvalue, $\omega = 0.0726 + 0.0549i$, and eigensolution of the harmonic Beltramian bidirectional vortex with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. 251
6.36 Axial wave contour for the first unstable eigenvalue $N = 50$, $q = 1$, $Re = 2,000$, and $\kappa = 0.1$. 253
6.37 Figure by Chedevergne [1] illustrating the effect of chamber length compared with current results for the complex-lamellar BV. 255

7.1 Asymmetric spectral plots of the Taylor-Culick mean flow, with $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$. 259
7.2 A comparison between the LNP and biglobal approach applied to the complex-lamellar mean flow with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. 260
7.3 Comparison between the LNP and biglobal asymmetric axial velocity contour plots, with $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$. 261
7.4 Comparison between the LNP and biglobal asymmetric pressure contour plots, with $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$. 262
7.5 A comparison between the spectra of the linear Beltramian model to identify the effect of a swirl component in the mean flow. Here, $\alpha = 3$, $z = 2$, $Re = 10,000$, and $\kappa = 0.1$. 263

7.6 Comparison between the linear Beltramian axial wave contour plots to illustrate the waveform when U_θ is omitted. Here, $\alpha = 0.5$ $Re = 10,000$, and $\kappa = 0.1$. The axisymmetric biglobal waveform is zero. 265

7.7 The multidirectional, linear Beltramian model for two values of κ. Here $q = 1$, $Re = 10,000$, and $l = 2$. 266

7.8 Comparison between the multidirectional linear Beltramian axial wave contour plots for two values of κ. Here $q = 1$, $Re = 10,000$, and $l = 2$. 267

Nomenclature

A_{ij}	=	the operator matrix
B	=	tangential angular momentum, rU_θ
B_i	=	the forcing function column vector
B_{ij}	=	the right hand side operator matrix of a matrix pencil
D_N	=	the Chebyshev pseudo-spectral derivative matrix of size N over the domain $[-1, 1]$
$\bar{D}_N$	=	the Chebyshev pseudo-spectral derivative matrix of size N mapped over arbitrary domain
H	=	total nondimensional stagnation pressure head, $p + \frac{1}{2}u^2$
I_N	=	the identity matrix of size N
L	=	dimensional chamber length
M	=	denotes a base flow component
$\tilde{M}$	=	denotes an instantaneous flow component
P	=	denotes a base flow pressure
$\tilde{P}$	=	denotes a instantaneous flow pressure
Q_i	=	nondimensional inlet volumetric flow rate, $\bar{Q}_i/(Ua^2)$
$\bar{Q}_i$	=	dimensional inlet volumetric flow rate
Re	=	Reynolds number
S	=	swirl number, $\pi ab/A_i = \pi\beta\sigma$
T_N	=	a Chebyshev polynomial of the first kind
U	=	tangential injection velocity at $\bar{\boldsymbol{U}}(a, L)$
U_N	=	a Chebyshev polynomial of the second kind
$\boldsymbol{U}$	=	base flow velocity vector, $U_r\boldsymbol{e}_r + U_\theta\boldsymbol{e}_\theta + U_z\boldsymbol{e}_z$

$\tilde{U}$	=	denotes the instantaneous flow velocity
V	=	vortex Reynolds number, $V = 2\pi\kappa/\varepsilon$
a	=	dimensional chamber radius
b	=	dimensional exit port radius
d	=	weighting coefficients for pseudo-spectral derivative matrices
l	=	chamber aspect ratio
m	=	denotes a general amplitude function
$\breve{m}$	=	denotes an hydrodynamic unsteady fluctuation
$\hat{m}$	=	denotes an acoustic unsteady fluctuation
m'	=	denotes a general unsteady fluctuation
$\tilde{m}$	=	denotes an vortical unsteady fluctuation
p	=	denotes a pressure amplitude function
$\breve{p}$	=	denotes an hydrodynamic pressure fluctuation
$\hat{p}$	=	denotes an acoustic pressure fluctuation
p'	=	denotes a general unsteady pressure fluctuation
$\tilde{p}$	=	denotes an vortical pressure fluctuation
q	=	the azimuthal integer wave number
r	=	radial nondimensional coordinate
u	=	denotes a velocity amplitude function
$\breve{u}$	=	denotes an hydrodynamic velocity fluctuation
$\hat{u}$	=	denotes an acoustic velocity fluctuation
u'	=	denotes a general unsteady velocity fluctuation
$\tilde{u}$	=	denotes a vortical velocity fluctuation
x	=	streamwise spatial variable over arbitrary domain

y	=	transverse spatial variable over arbitrary domain
z	=	axial nondimensional coordinate

Subscripts

N	=	Chebyshev polynomial order/number of collocation points
i	=	denotes the matrix row or vector element
ii	=	denotes a diagonal matrix or diagonal element
ij	=	denotes a matrix
j	=	denotes the matrix column
m	=	corresponds to the mth root of the Bessel function, $J_1(x)$
o	=	denotes an outlet condition
r	=	denotes the radial spatial direction
θ	=	denotes the tangential spatial direction
z	=	denotes the axial spatial direction
0	=	corresponds to the first root of the Bessel equation, $J_1(x)$

Superscripts

(c)	=	a composite solution
(i)	=	a solution in the inner region, core or wall
n	=	denotes the order of the derivative
$\bar{}$	=	denotes dimensional variables

Symbols

Γ	=	circulation

∇	=	gradient operator
$\boldsymbol{\Omega}$	=	base flow vorticity vector, $\Omega_r \boldsymbol{e}_r + \Omega_\theta \boldsymbol{e}_\theta + \Omega_z \boldsymbol{e}_z$
α	=	longitudinal wave number
α	=	constant (Ch. 2 only), complex-lamellar: $\frac{1}{6}\pi^2 - 1$, Beltramian: $\frac{\lambda_0}{2\beta J_1(\lambda_0\beta)}\left(\frac{1}{8}\lambda_0^2 - 1\right)$
β	=	normalized outlet radius, b/a
γ	=	constant, $\frac{\lambda_0}{4\beta J_1(\lambda_0\beta)}$
δ	=	denotes the characteristic boundary layer thickness
ε	=	$1/Re$
η	=	streamwise Chebyshev variables mapped between $[-1, 1]$
κ	=	inflow parameter, $Q_i/(2\pi l) = (2\pi\sigma l)^{-1}$
λ	=	eigenvalue
λ_m	=	root of the Bessel function, $J_1(x)$
ν	=	kinematic viscosity
ω	=	frequency of oscillation, $\omega_r + \omega_i$
ω_r	=	real component of ω, circular frequency
ω_i	=	imaginary compoment of ω, growth rate
$\tilde{\omega}$	=	distributed growth rate
θ	=	tangential nondimensional coordinate
ψ	=	streamfunction
σ	=	modified swirl number, $Q_i^{-1} = S/(\pi\beta)$
ξ	=	dependent variable transformation (Ch. 2 only)
ξ	=	radial Chebyshev variables mapped between $[-1, 1]$

Abbreviations

BG	=	biglobal
BV	=	bidirectional vortex
DNS	=	direct numeric simulation
LNP	=	local nonparallel
LNS	=	linearized Navier-Stokes
NNP	=	normal nonparallel
NPR	=	nonparallel ratio
ODE	=	ordinary differential equation
OSE	=	Orr-Sommerfeld equations
PDE	=	partial differential equation
PSE	=	parabolized stability equations
VCCWC	=	Vortex Combustion Cold-Wall Chamber
VIHRE	=	Vortex Injection Hybrid Rocket Engine

Nondimensionalization

$$z = \frac{\bar{z}}{a}; \quad r = \frac{\bar{r}}{a}; \quad \nabla = a\bar{\nabla}; \quad t = \bar{t}\frac{U}{a};$$
$$\tilde{U}_r = \frac{\bar{U}_r}{U}; \quad \tilde{U}_\theta = \frac{\bar{U}_\theta}{U}; \quad \tilde{U}_z = \frac{\bar{U}_z}{U};$$
$$\tilde{P} = \frac{\bar{P}}{\rho U^2}; \quad Q_i = \frac{\bar{Q}_i}{Ua^2}; \quad l = \frac{L}{a}; \quad Re = \frac{Ua}{\nu}$$

Chapter 1

Introduction

1.1 The Bidirectional Vortex

Characterization of unidirectional vortex flows has remained a central topic of interest in fluid dynamics since its earliest beginnings. The earliest work in print is that of Rankine [2] in 1858. Other notable advancements may be attributed to Lamb-Oseen [3, 4], Burgers-Rott [5, 6], Batchelor [7], and others. These original studies are still relevant in a variety of modern applications. For example, the Rankine vortex is still used as a crude approximation for describing the bulk motion of hurricanes and other large, atmospheric, swirl-dominated flows. Both Lamb-Oseen and Burgers-Rott models can be applied to localized atmospheric swirling flows such as tornados, dust devils, and water spouts [8]. Other recent investigations by Alekseenko *et al.* [9], Eloy and Le Dizs [10], Schmid and Rossi [11], Olendraru and Sellier [12], Prez-Saborid *et al.* [13], and others reveal the continued interest in these vortical flows.

Sullivan's 1959 solution of an external two-cell vortex [14] may be the first evidence of bidirectional vortex modeling. Its usefulness is slightly overshadowed by its characterization in terms of integral functions. Numerical integration clearly shows the existence of two fundamental regions common to all vortex models: a forced vortex forming around the axis

of rotation and a free, nearly irrotational, vortex tail. While the free vortex is inviscid, the character of the forced vortex is dominated by viscous stresses. The distinct inner downflow of the Sullivan vortex is often seen in tornados and possesses considerable relevance in meteorological studies [15].

Bloor and Ingham analyzed the conical flow in cyclonic separators. Their solution, albeit inviscid, may be considered a milestone achievement in advancing the theory of confined swirl dominated flows. In their 1987 paper [16], they introduce an inviscid approximation to a conical vortex chamber in the context of cyclonic seperators. Their solution is later reconstructed and extended by Barber and Majdalani [17]. Both solutions, being inviscid, displayed a centerline singularity while also allowing slip at the sidewall.

Bidirectional motion was first proposed by Gloyer, Knuth and Goodman [18] as a means to enhance the burn-rate in hybrid rocket engines. Later, this technology was furthered by Knuth *et al.* [19], Chiaverini *et al.* [20], and others in the development of the Vortex Injection Hybrid Rocket Engine (VIHRE) and the Vortex Combustion Cold-Wall Chamber (VCCWC) shown in Fig. 1.1. The latter refers to an internally cooled liquid thrust chamber driven by a pair of coaxial vortices. The inception of the VCCWC also spurred research by Vyas and Majdalani [21] and later Majdalani and Rienstra [22] to develop the first of three analytical models describing its fully three-dimensional motion [21]. Singularities encountered by Bloor and Ingham [16] also appeared in their purely inviscid profile. Some of these limitations were remedied by the use of viscous approximations to the core vortex and tangential sidewall shear layer [23]. Shortly thereafter, Batterson and Majdalani [24] completed the viscous treatment by incorporating viscous wall corrections in the remaining axial and radial directions.

Soon following the complete characterization of the first, complex-lamellar model, Majdalani [26] utilized the incompressible Bragg-Hawthorne equation to produce both the linear and harmonic Beltramian vortex solutions that display linear and sinusoidal streamwise dependence, respectively. Solutions of the Beltramian type are characterized by a

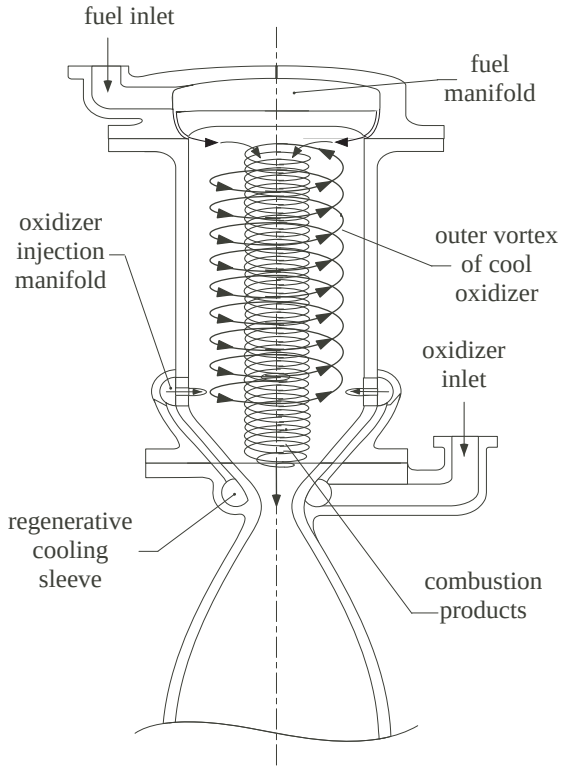


Figure 1.1: VCCWC schematic [25].

zero Lamb vector, $\bar{\boldsymbol{\omega}} \times \bar{\boldsymbol{u}} = 0$, to the extent that their vorticity and velocity remained directly proportional through $\bar{\boldsymbol{\omega}} = \lambda_0 \bar{\boldsymbol{u}}$ [27]. Again, these models shared the inviscid limitations in regions where fluid friction could not be ignored. Batterson and Majdalani brought closure to the Beltramian solutions by characterizing the shear layers throughout the chamber [28, 29]. The analysis for the Beltramian viscous layers closely mimicked that of the previous complex-lamellar study. Certain differences were characterized and directly attributed to the effect of the unique base flow.

The presence of multi-directional vortex motion was first seen as an artifact of higher eigensolutions stemming from Vyas and Majdalani's complex-lamellar model [30]. However, experimental and numerical evidence gathered by Anderson *et al.* [31], Rom, Anderson and Chiaverini [32], and others have corroborated the existence of interchanging flow

reversals. Initial observations suggested excellent agreement between the experimental, numerical, and complex-lamellar results for an even number of flow reversals (four). Further investigation along with the development of the Beltramian solutions suggested that the initial comparisons with the experimental data was misguided and that only an odd number of flow reversals was physical. The recent work by Batterson and Majdalani [33] provided a more suitable comparison where the Beltramian solutions, rather than the complex-lamellar, are used to predict the locations of internal mantles associated with each flow reversal.

The key findings from the complex-lamellar and Beltramian bidirectional vortex solutions can be found in Ch. 2.

1.2 Instability

Instability research stands at the forefront of many modern real-world problems. Hydrodynamic instability has been in development for well over a century and has fostered groundbreaking work in many facets of engineering and mathematical research. The pioneers in the field include names like Helmholtz [34], Kelvin [35], and Rayleigh [36]. Perhaps the most widely known work is the laminar-turbulent transition experiments published by Reynolds in 1883 [37]. Therein, Reynolds observed and recorded the conditions for which a stream of dye entrained in a pipe flow changes from a distinct stream to a uniform mixture with the surrounding flow. Since then, transitional flow has become a science unto itself with pertinent applications in aerodynamic flow separation, lift and drag, and even cyclical loading in aerodynamic structures.

Vortex shedding has been identified as a significant mechanism affecting instability and the transition from laminar to turbulent flow. It leads directly to turbulence as well as a tripping mechanism to initiate the formation of hydrodynamic waves. Three types of vortex shedding are described in Fig. 1.2 [38]. The first, known as obstacle vortex shedding, occurs when vortices are spun off of the trailing edge of a protrusion into the flow. Angle vortex

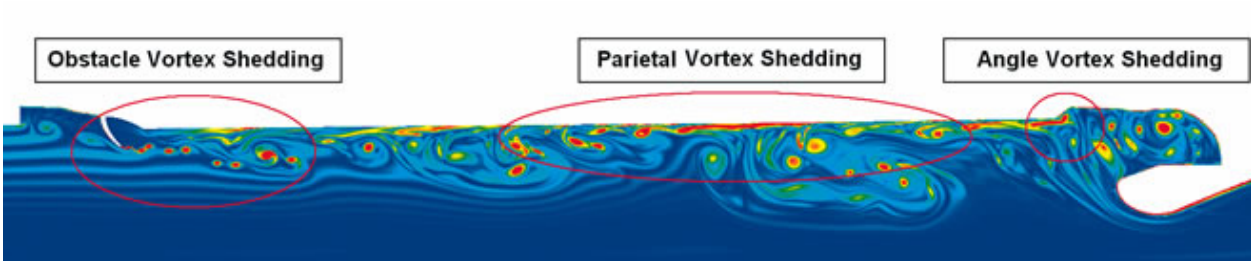


Figure 1.2: The three identified types of vortex shedding [38].

shedding is characteristic of a backward facing step in which the flow tries to accommodate a sharp turning angle. Lastly, parietal vortex shedding, coming from the French word 'paroi' meaning wall, is a product of fluid/wall interaction and is an intrinsic instability of many mean flowfields. This type of vortex shedding is an inherent hydrodynamic instability and acts as a primary mechanism in laminar/turbulent transition. All three types can lead to serious complications in thrust chambers. As the name suggests, parietal vortex shedding is characterized by oscillations where the largest amplitudes occur near the surface and diminish near the centerline. Parietal oscillations associated with the vortico-acoustic wave are attributed to a vortical boundary layer generated by a reciprocating longitudinal acoustic wave. Its presence stands to fulfill the no-slip condition on the precluding acoustic wave. Although nontrivial, this unsteady behavior can be characterized analytically for many closed-form base flow profiles by employing a series of exact or asymptotic mathematical techniques. Hydrodynamic wave formation is due to bulk flow vorticity generation. Here, the no-slip condition is self-satisfied by enforcing the 'zero' condition at the boundaries. As such, this formulation is discussed as a generalized eigenvalue problem and must be solved numerically. Its solution is addressed by spectral methods where the eigenvalues represent the circular frequency and the eigenvectors capture the flowfield.

Generally, efforts are taken to reduce and eliminate instability in propulsive applications. Although the argument could be made that increased flow oscillation and turbulent breakdown increases combustion efficiency as a avenue toward increased localized mixing of

fuel and oxidizer, the costs outweigh the benefits. Thrust oscillations cause fatigue material failure in the chamber. Oscillations often occur near a natural harmonic of the combustion chamber and then resonate throughout the entire vehicle. Instability is often accompanied by an increase in mean pressure, known as the "DC shift." This sustained increase in pressure can exceed the maximum hoop stress in the chamber if not properly taken into account. Vibrational loading can also cause structural failure in other systems as well. Secondary failures, such as guidance system and targeting failure, are common when unpredicted thrust oscillations appear. The problem is exacerbated where oscillations can exceed safe g-loading for manned missions.

The mathematical and physical ideas associated with instability have been collected from the areas of fluid dynamics and heat transfer. The well known techniques based on the normal mode approach were already well developed in the realm of dynamical systems, particle mechanics, and rigid body motion before coming to light in what is now hydrodynamic instability [39]. Though sound in principle, rigorous analytic solutions are not easily recovered due to the complexity of realistic mean flowfields. Exact solutions of the steady inviscid Euler or the viscous Navier-Stokes equations form the basis of the mathematical instability problem. Since few analytic solutions exist for complex geometries that would be indicative of obstacle and angle vortex shedding, parietal instability of simple flow models becomes the frontrunner for analytic research. The fundamental understanding of this type of instability can naturally be applied to understand the effect of the other two. In light of increasing computational power, CFD has been successfully used to calculate the base, or mean, flowfield in which instability analysis is then applied. Theofilis and Sherwin have successfully applied stability analysis to their computationally resolved base flow along the trailing edge of an airfoil [40]. In their work, Theofilis and Sherwin remark that numerical residuals at $\mathcal{O}(1)$ become forcing functions in the higher order instabilities [41]. Along similar lines, Barkley *et al.* use spectral element discretization to obtain the two-dimensional base

flow solution over a backward facing step [42]. Likewise, the base flow around a forward facing step is determined via finite-volume methods by Stuer and co-workers [43, 44].

Linear stability analysis has produced verifiable results in many practical applications. The method of application is relatively straightforward. We superimpose a higher order fluctuating term over the steady base flow. The resulting instantaneous velocity becomes of the form $\tilde{M} = M + m'$, with $\tilde{M}$ representing the instantaneous flow: the sum of the base flow, M, and a corresponding unsteady fluctuation, m'. The Navier-Stokes equations are perturbed by inserting the instantaneous velocity. Assuming knowledge of an analytical or computational base flow a priori, the terms consisting of only the base flow can be assumed to satisfy the leading order solution and are negated from the equation of motion. Also, terms of $\mathcal{O}(m'^2)$ are truncated. The remaining terms comprise the Linearized Navier-Stokes (LNS) equations. The LNS equations denote a set of linear coupled partial differential equations governing the fluctuation in which the remnants of the base flow appear as variable coefficients.

1.2.1 On the Vortico-Acoustic Wave Formulation

The mathematical formulation and solution of the vortico-acoustic wave equations is well understood beginning with the extensive studies of the Stokes boundary layer with nontranspiring walls [45–48]. The increased complexity incurred when considering wall injection was thoroughly addressed by Majdalani [49]. While the solution to the one-dimensional acoustic component is well known, he successfully employed asymptotic techniques to resolve the otherwise intractable equation for the solenoidal flow component. As is now the convention, the distinguishing parameters were used to reduce the intractable exact equation to a system of recursive tractable equations. Concurrently, Flandro employed a different method to the same problem based on the vorticity transport equation [50]. His approach was later refined and compared to Majdalani's model [51].

Presently, the formulation of the vortical component to the problem is well understood. The solution, however, is difficult. Conventional numeric schemes fail for two major reasons. First, for high Reynolds number flows the equations become stiff and cause numeric instability. Second, the centerline boundary condition is difficult to secure with standard shooting methods. Successful attempts have shown that the unsteady vorticity wave deteriorates to a value near, but not exactly, zero near the centerline. Through numerical experimentation, we find convergence to be difficult to achieve for many input parameters. The stiff nature of the equation can be seen to cause numeric instablity as well as extreme sensitivity in the shooting criteria. In essence, sizeable changes in the wave form are observed for minuscule changes in the shooting trajectory. For certain parameters and numerical schemes, it can be shown that convergence can only be accomplished when changes in the shooting trajectory are kept extremely small (10^{-16}); specifically, below the code's precision.

Asymptotic solutions for vortical wave problems with both suction and injection for a myriad of geometries are now available [49, 51–54]. Simple flow profiles lend themselves to multiple scales and, in some cases, exact solutions while more general solutions have only been rendered using WKBJ or generalized scaling methods introduced by Majdalani [54].

1.2.2 On the Hydrodynamic Wave Formulation

Hydrodynamic instability has breached the gap between external flows in aerodynamics and internal flows in propulsion. Though the formulation for both external and internal flows is primarily the same, the motivation for research is significantly different. Instead of flow separation and transition to turbulence being the forefront of stability research, internal flow analysis is focused on another aspect of hydrodynamic instability: the hydrodynamic wave. At the heart of classic linear stability lies a one-dimensional normal mode analysis. The

fluctuating components take on the ansatz*

$$\breve{m}\left(r,\theta,z,t\right)=m\left(r\right)\exp\left[i\left(q\theta+\alpha z-\omega t\right)\right] \tag{1.1}$$

where $m(r)$ is a one-dimensional amplitude function, α is a complex wave number, ω is the complex circular frequency, and q is the an integer mode number. The exponential term captures the oscillatory nature and the growth of wave amplitude in both space and time. Direct substitution of this ansatz into the LNS equations, in which only derivatives with respect to r are kept, recovers a system of coupled ordinary differential equations known as the Orr-Sommerfeld equations (OSE) (see Drazin [39]). A closer look at this approach reveals significant limitations of the Orr-Sommerfeld equations to general flow problems. The applicability of this type of analysis is determined by the parallel-flow assumption. This requirement states that the base flow must be a function of the radial component only. In other words $\frac{\partial M}{\partial z}=\frac{\partial M}{\partial \theta}=0$. Simple Poiseuille and Couette flow do indeed adhere to this criterion; however, more interesting flows such as the Taylor plane flow or the Taylor-Culick axisymmetric flow do not.

Two hydrodynamic theories are born: spatial and temporal theories [55]. Spatial theory allows growth and decay only in space by setting the imaginary part of the frequency to zero. Conversely, temporal theory allows only growth in time by setting the imaginary component of the wavenumber to zero. Gaster has shown that both theories should produce the same results within a small truncation error by integrating the Cauchy-Riemann relations, $\partial\omega_r/\partial\alpha_r=\partial\omega_i/\partial\alpha_i$ and $\partial\omega_r/\partial\alpha_i=-\partial\beta_i/\partial\alpha_r$ [56].

The restriction of the parallel-flow assumption can be relaxed by suggesting that $\frac{\partial M}{\partial z},\frac{\partial M}{\partial \theta}\ll\frac{\partial M}{\partial r}$. In doing so, we are able to extend the normal mode approach to a much larger range of base flows. This is justified when the parallel-flow assumption is nearly satisfied locally. This technique is widely known as the Local Nonparallel (LNP) or Normal

*$\breve{m}\left(y,x,z,t\right)=m\left(y\right)\exp\left[i\left(\alpha x+\beta z-\omega t\right)\right]$

Nonparallel (NNP) approach. A comprehensive study regarding the application of the LNP approach to solid rocket motors has been presented by Casalis, and Vuillot [55]. Their work shows excellent agreement with experimental data and suggests that the LNP technique is a viable predictor of stability in long chambers.

Though the LNP technique has been successfully applied to long solid rocket models, the localized parallel-flow assumption is not entirely justified. For a two-dimensional base flow, the axial velocity is related to a nonzero transverse velocity through continuity. This is why the axial dependence is considered nonparallel [55]. The degree of nonparallelism can be qualitatively determined by calculating the magnitude of the ratio of the transverse-to-axial velocities. In the case of the Taylor-Culick profile in solid rocket motors, this ratio asymptotes to zero downstream. This behavior produces a viable justification for its use in long chambers along the centerline. However, in the headwall region and near the sidewall the local parallel-flow assumption remains in question. Also, since the LNP is based on a one-dimensional amplitude function, it returns a system of ordinary differential equations. Consequently, it cannot satisfy the boundary conditions at the headwall.

Griffond has shown the LNP approach to be mathematically inconsistent [57]. In his analysis, he shows that the governing equations, and inevitably the solutions, are formulation dependent. Converting a primitive variable formulation of the governing equations to a streamfunction representation leads to an extra term not shown in a direct streamfunction derivation. Strictly enforcing the parallel flow assumption in either formulation results in the emergence of the Orr-Sommerfeld equations. He attributes this deficiency to the inability of both models to satisfy the parallel-flow assumption.

A third approach in the realm of classic linear theory is that of the Parabolized Stability Equations (PSE) [58]. It was originally developed to reconcile the questions remaining in the stability of the Blasius boundary layer. Similar to the LNP approach, the formulation begins with the relaxation of the parallel-flow assumption. Unlike the LNP however, the

ansatz takes the form

$$\breve{m}\left(r,\theta,z,t\right)=m\left(r,z\right)\exp\left[\int_{z_0}^{z}i\alpha\left(\xi\right)\,\mathrm{d}\xi+q\theta-i\omega t\right] \tag{1.2}$$

This method allows for spatial variation of the wavenumber and forces two-dimensional dependence on the amplitude function. Its two-dimensionality permits the PSE approach to remain quasi-consistent [59]. Instead of the standard eigenvalue problem produced by the LNP technique, the PSE approach results in stepped calculations in the streamwise direction [60]. Regardless, it is shown by Casalis and Vuillot [55] that the PSE and LNP techniques are in excellent agreement with each other and experimental data for simple base flows.

The difference in the PSE approach from the standard eigenvalue problem leads to an interesting, but not obvious difficulty with the formulation. Haj-Hariri [61] has analyzed the mathematical form of the PSE approach and determined that it resulted in a system of weakly elliptic, rather than parabolic, equations. In his assessment, this renders an explicit scheme in the streamwise direction numerically unstable. Researchers are forced to use implicit methods to overcome this deficiency with a large enough step size in the streamwise direction to suppress the numerical artifacts. Li and Malik address this issue in 1995 and 1996 where they discuss the critical step size [62, 63] needed for convergence. This limitation leads to a balance of the streamwise accuracy and the numeric stability of the solution. Andersson and co-workers suggest using a numeric stabilizing technique by first approximating the streamwise derivative with a first order implicit scheme and including a stabilizing term at the order of the truncation error [64]. Their techniques have shown a significant increase in numerical stability.

It should be noted that the relaxed form of the parallel flow assumption is the foundation of this formulation. For strongly nonparallel flows or nonparallel regions, the validity is still in question. Also, by allowing the wavenumber to vary, the PSE formulation results in an ill-posed system of equations. An auxiliary equation known as the normalization condition

must be introduced. This equation has two purposes: to produce a well-posed problem and to shift as much streamwise dependence as possible into the exponential. The latter purpose supplements the relaxation of the parallel flow assumption by forcing the amplitude function to slowly vary in the second spatial dimension. The normalization condition appears in many forms and is often problem dependent. Nonetheless, it produces a viable closure to the PSE system [58].

Because the integration of the wave number begins at the neutral curve, the initialization of the problem is very important. It must be initialized in the stable flow within an acceptable proximity to the neutral curve. If the starting point is too close, transient characteristics may not be resolved. If too far, an unrealistically large transient region may appear. Likewise, the onset of stability may be shifted upstream or downstream if the initialization is premature or tardy. Previous knowledge of the location of the neutral curve can be determined from the simplified OSE calculations; however, the introduction of nonparallel effects can shift the neutral curve significantly upstream [60]. Therefore, ambiguity may be seen to influence the ensuing equations.

The PSE and LNP equations can be written to distinctly show how the OSE equation takes the role of the leading order solution, with streamwise variations falling into first order and higher approximations [55, 65, 66]. We have already stated that the use of the Orr-Sommerfeld equations are not valid because of the parallel-flow assumption for flows such as the Taylor-Culick profile. Therefore a universally consistent, rigorous approach needs to be sought to accommodate real flow characteristics at all orders.

The latest advances in hydrodynamic instability have led us to what is known as biglobal hydrodynamic instability. The idea of modal decomposition continues through this formulation. In the biglobal context, the ansatz takes yet another form namely,

$$\breve{m}(r,\theta,z,t) = m(r,z)\exp\left[i\left(q\theta - \omega t\right)\right] \tag{1.3}$$

This form allows for significant variations in the axial and radial directions and periodicity in the tangential. By allowing the amplitude function to be two-dimensional, we recover a fully consistent formulation and a well-posed system. Another advantage of a two-dimensional approach is that the problematic parallel-flow assumption is eliminated. This allows rigorous solutions to be calculated for a large number of axisymmetric and two-dimensional flows. Like the PSE equations, this formulation produces a system of partial differential equations rather than simply the ordinary differential equation sets created by the OSE and LNP approaches. In doing so, it allows for the satisfaction of no-slip at the headwall. Chedevergne and co-workers [1] use a linear extrapolation as suggested by Lin and Malik [67] and Theofilis [41] to deduce boundary conditions at the exit plane. DNS simulation suggests that this formulation produces accurate results for arbitrary exit plane locations.

The partial differential equation formulation leads to added complexity in the solution of the biglobal eigenvalue problem. Spectral collocation methods and Arnoldi eigensolvers currently present the most efficient means of computation. Rather than the powerful QZ algorithm [68, 69] which captures all the eigenvalues at once, Arnoldi iteration schemes [70–72] efficiently calculate eigenvalues in a user specified region. The full spectrum can be solved by repeating the Arnoldi iteration in overlapping regions. In doing so, each run requires much less storage, memory, and CPU power to produce accurate results.

Theofilis comments that the overall power of biglobal stability predictions is slightly tempered by its inability to directly handle spatial theory [41]. He states:

> *"If two spatial directions are resolved, the eigenmodes will not possess a wave-like character in the general case [73] and it is not clear whether the concept of a complex wavenumber can be simply borrowed from its counterpart terminology in one-dimensional linear analysis...*
>
> *...Furthermore, from a numerical point of view Heeg and Geurts [74] have documented that the computing effort of a spatial biglobal eigenvalue problem*

solved with the Jacobi-Davidson algorithm scales with the size of the problem solved raised to a power between two and three; this could become excessive compared with the cost of the respective temporal problem."

More concisely, Theofilis states that a *sinusoidal* spatial wave cannot be expected since the spatial waveform is no longer specified in the ansatz; rather, it is defined in a two-dimensional amplitude function. In any event, this perspective may be inconsequential given that the two-dimensional eigenmode is sufficient to describe the spatial evolution of the fluctuations.

The development of biglobal hydrodynamic instability theory has opened the door to confidently explore increasingly complex flowfields. It is apparent that the alleviation of the parallel flow assumption through biglobal stability is a significant advancement over classic theory. With its successful application and validation as a rigorous and accurate method to predict temporal laminar/turbulent transition and wave propagation, it will continue to be a pertinent tool for hydrodynamic instability research.

1.2.3 On Linear Hydrodynamic Stability Theory

The instantaneous velocity is considered to be the sum of a steady[†] and three oscillating perturbations

$$\tilde{M} = M + \hat{m} + \tilde{m} + \breve{m} \tag{1.4}$$

$\tilde{M}$ is the instantaneous flow component. Next, $\hat{m}$, is the compressible, irrotational acoustic wave. The rotational, incompressible vortical wave is represented by $\tilde{m}$. These two are inherently related in that $\tilde{m}$ does not exist without the $\hat{m}$. The sum of these two waves satisfies the no-slip condition at the wall. Both the wavelengths and frequencies of these two waves are discernable. Also, the vortical wave is parietal such that its largest magnitudes

[†]In this context, "steady" is synonymous with non-oscillatory. The flow can be consistently varying in time but is considered "steady" since the temporal variation of the perturbation is much faster than the base flow.

are exhibited near the wall. Finally, the hydrodynamic wave is given as $\breve{m}$. It is also a parietal wave; however, because of its connection with turbulence, it occurs over a spectrum of frequencies and scales. This instability automatically satisfies the no slip condition at the wall without the need for additional boundary layer corrections. Thus, spectral methods must be used to test the range of frequencies and determine the most amplified modes.

The assumption is made here that the hydrodynamic instability waves remain uncoupled from any of the others and therefore can be summed in a linear fashion. This is because each wave occurs over unique scales. Speculation exists as to the validity of this assumption [75]. Since the hydrodynamic wave essentially occurs over a range of scales, it is likely to couple with the vortico-acoustic wave at some frequencies. However traditional thinking has shown viable results. This assumption allows us to make progress by analyzing each wave independently from the others.

The analysis begins with the superposition of the base flow at leading order with the hydrodynamic wave being represented as a first order perturbation.

$$\tilde{M} = M + \breve{m} \tag{1.5}$$

In strictly mathematical terms, the following shows a pure mathematical derivation to obtain a linear relation from the Navier-Stokes equations. If we express the complete set of equations together in terms of a nonlinear operator $\mathcal{N}$, then the condition on $\breve{m}$ can easily be identified [55]:

$$\mathcal{N}(\tilde{M}) = 0 \rightarrow \mathcal{N}(M + \breve{m}) = 0 \tag{1.6}$$

However, we must assume a known solution for the base flow can be determined analytically or computationally a priori. Therefore the base flow alone already satisfies

$$\mathcal{N}(M) = 0 \tag{1.7}$$

This is expressed explicitly through a Taylor expansion

$$\mathcal{N}(M + \breve{m}) = \mathcal{N}(M) + \mathcal{L}(M) \cdot \breve{m} + \mathcal{O}(\breve{m}^2) = 0 \tag{1.8}$$

where we identify the new *linear* operator, $\mathcal{L}(M)$, acting on $\breve{m}$. Since $\mathcal{N}(M) = 0$ is assumed to be known a priori and terms of $\mathcal{O}(\breve{m}^2)$ are truncated, we are left only with the first order equation for the hydrodynamic fluctuation; namely, $\mathcal{L}(M) \cdot \breve{m} = 0$. In a strictly perturbative sense each order is, by definition, identically zero. Applying these concepts to continuity and the Navier-Stokes equations leads us to the *Linearized Navier-Stokes (LNS)* equations.

The operator description of the LNS equations given above is general and can encompass the complete Navier-Stokes equation including rotationality, compressibility, and viscosity. Conversely, it works for any physical assumptions used to reduce the full equation. The general hydrodynamic equations are derived under the auspices of rotational, viscous, incompressible flow. Thus, the hydrodynamic LNS equations can be formulated by considering the instantaneous continuity and momentum equations of the form

Continuity:

$$\frac{\partial \tilde{U}_r}{\partial r} + \frac{\tilde{U}_r}{r} + \frac{1}{r}\frac{\partial \tilde{U}_\theta}{\partial \theta} + \frac{\partial \tilde{U}_z}{\partial z} = 0 \tag{1.9a}$$

Radial momentum:

$$\begin{aligned}\frac{\partial \tilde{U}_r}{\partial t} + \tilde{U}_r\frac{\partial \tilde{U}_r}{\partial r} + \frac{\tilde{U}_\theta}{r}\frac{\partial \tilde{U}_r}{\partial \theta} - \frac{\tilde{U}_\theta^2}{r} + \tilde{U}_z\frac{\partial \tilde{U}_r}{\partial z} + \frac{\partial \tilde{P}}{\partial r} \\ = \frac{1}{Re}\left(\frac{\partial^2 \tilde{U}_r}{\partial r^2} + \frac{1}{r}\frac{\partial \tilde{U}_r}{\partial r} - \frac{\tilde{U}_r}{r^2} + \frac{1}{r^2}\frac{\partial^2 \tilde{U}_r}{\partial \theta^2} - \frac{2}{r^2}\frac{\partial \tilde{U}_\theta}{\partial \theta} + \frac{\partial^2 \tilde{U}_r}{\partial z^2}\right)\end{aligned} \tag{1.9b}$$

Tangential momentum:

$$\frac{\partial \tilde{U}_\theta}{\partial t} + \tilde{U}_r \frac{\partial \tilde{U}_\theta}{\partial r} + \frac{\tilde{U}_\theta}{r}\frac{\partial \tilde{U}_\theta}{\partial \theta} + \frac{\tilde{U}_r \tilde{U}_\theta}{r} + \tilde{U}_z \frac{\partial \tilde{U}_\theta}{\partial z} + \frac{1}{r}\frac{\partial \tilde{P}}{\partial \theta} = \frac{1}{Re}\left(\frac{\partial^2 \tilde{U}_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial \tilde{U}_\theta}{\partial r} - \frac{\tilde{U}_\theta}{r^2} + \frac{1}{r^2}\frac{\partial^2 \tilde{U}_\theta}{\partial \theta^2} + \frac{2}{r^2}\frac{\partial \tilde{U}_r}{\partial \theta} + \frac{\partial^2 \tilde{U}_\theta}{\partial z^2}\right) \quad (1.9c)$$

Axial momentum:

$$\frac{\partial \tilde{U}_z}{\partial t} + \tilde{U}_r \frac{\partial \tilde{U}_z}{\partial r} + \frac{\tilde{U}_\theta}{r}\frac{\partial \tilde{U}_z}{\partial \theta} + \tilde{U}_z \frac{\partial \tilde{U}_z}{\partial z} + \frac{\partial \tilde{P}}{\partial z} = \frac{1}{Re}\left(\frac{\partial^2 \tilde{U}_z}{\partial r^2} + \frac{1}{r}\frac{\partial \tilde{U}_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \tilde{U}_z}{\partial \theta^2} + \frac{\partial^2 \tilde{U}_z}{\partial z^2}\right) \quad (1.9d)$$

These equations are perturbed by defining the instantaneous velocity as the linear sum of the base flow and a fluctuation attributed to the hydrodynamic wave. In general terms we define

$$\tilde{M} = M + m', \qquad m' = \hat{m} + \tilde{m} + \breve{m} \quad (1.10)$$

corresponding to the base flow and sum of oscillating perturbations. For hydrodynamic instability we Here, $m' = \breve{m}$ only. This step ends with

Continuity:

$$\frac{\partial U_r}{\partial r} + \frac{U_r}{r} + \frac{1}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{\partial U_z}{\partial z} + \frac{\partial \breve{u}_r}{\partial r} + \frac{\breve{u}_r}{r} + \frac{1}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\partial \breve{u}_z}{\partial z} = 0 \quad (1.11a)$$

Radial momentum:

$$
\begin{aligned}
&\frac{\partial U_r}{\partial t} + U_r\frac{\partial U_r}{\partial r} + \frac{U_\theta}{r}\frac{\partial U_r}{\partial \theta} - \frac{U_\theta^2}{r} + U_z\frac{\partial U_r}{\partial z} + \frac{\partial P}{\partial r} \\
&\quad + \frac{\partial \breve{u}_r}{\partial t} + U_r\frac{\partial \breve{u}_r}{\partial r} + \breve{u}_r\frac{\partial U_r}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_r}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_r}{\partial \theta} - \frac{2U_\theta \breve{u}_\theta}{r} + U_z\frac{\partial \breve{u}_r}{\partial z} + \breve{u}_z\frac{\partial U_r}{\partial z} + \frac{\partial \breve{p}}{\partial r} \\
&\qquad + \breve{u}_r\frac{\partial \breve{u}_r}{\partial r} + \frac{\breve{u}_\theta}{r}\frac{\partial \breve{u}_r}{\partial \theta} - \frac{\breve{u}_\theta^2}{r} + \breve{u}_z\frac{\partial \breve{u}_r}{\partial z} \\
&\quad = \frac{1}{Re}\left(\frac{\partial^2 U_r}{\partial r^2} + \frac{1}{r}\frac{\partial U_r}{\partial r} - \frac{U_r}{r^2} + \frac{1}{r^2}\frac{\partial^2 U_r}{\partial \theta^2} - \frac{2}{r^2}\frac{\partial U_\theta}{\partial \theta} + \frac{\partial^2 U_r}{\partial z^2}\right. \\
&\qquad \left. + \frac{\partial^2 \breve{u}_r}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_r}{\partial r} - \frac{\breve{u}_r}{r^2} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_r}{\partial \theta^2} - \frac{2}{r^2}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\partial^2 \breve{u}_r}{\partial z^2}\right)
\end{aligned}
\tag{1.11b}
$$

Tangential momentum:

$$
\begin{aligned}
&\frac{\partial U_\theta}{\partial t} + U_r\frac{\partial U_\theta}{\partial r} + \frac{U_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r U_\theta}{r} + U_z\frac{\partial U_\theta}{\partial z} + \frac{1}{r}\frac{\partial P}{\partial \theta} \\
&\quad + \frac{\partial \breve{u}_\theta}{\partial t} + U_r\frac{\partial \breve{u}_\theta}{\partial r} + \breve{u}_r\frac{\partial U_\theta}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r \breve{u}_\theta}{r} + \frac{\breve{u}_r U_\theta}{r} + U_z\frac{\partial \breve{u}_\theta}{\partial z} + \breve{u}_z\frac{\partial U_\theta}{\partial z} + \frac{1}{r}\frac{\partial \breve{p}}{\partial \theta} \\
&\qquad + \breve{u}_r\frac{\partial \breve{u}_\theta}{\partial r} + \frac{\breve{u}_\theta}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\breve{u}_r \breve{u}_\theta}{r} + \breve{u}_z\frac{\partial \breve{u}_\theta}{\partial z} \\
&\quad = \frac{1}{Re}\left(\frac{\partial^2 U_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial U_\theta}{\partial r} - \frac{U_\theta}{r^2} + \frac{1}{r^2}\frac{\partial^2 U_\theta}{\partial \theta^2} + \frac{2}{r^2}\frac{\partial U_r}{\partial \theta} + \frac{\partial^2 U_\theta}{\partial z^2}\right. \\
&\qquad \left. + \frac{\partial^2 \breve{u}_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_\theta}{\partial r} - \frac{\breve{u}_\theta}{r^2} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_\theta}{\partial \theta^2} + \frac{2}{r^2}\frac{\partial \breve{u}_r}{\partial \theta} + \frac{\partial^2 \breve{u}_\theta}{\partial z^2}\right)
\end{aligned}
\tag{1.11c}
$$

Axial momentum:

$$\begin{aligned}\frac{\partial U_z}{\partial t} + U_r\frac{\partial U_z}{\partial r} + \frac{U_\theta}{r}\frac{\partial U_z}{\partial \theta} + U_z\frac{\partial U_z}{\partial z} + \frac{\partial P}{\partial z} \\ + \frac{\partial \breve{u}_z}{\partial t} + U_r\frac{\partial \breve{u}_z}{\partial r} + \breve{u}_r\frac{\partial U_z}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_z}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_z}{\partial \theta} + U_z\frac{\partial \breve{u}_z}{\partial z} + \breve{u}_z\frac{\partial U_z}{\partial z} + \frac{\partial \breve{p}}{\partial z} \\ + \breve{u}_r\frac{\partial \breve{u}_z}{\partial r} + \frac{\breve{u}_\theta}{r}\frac{\partial \breve{u}_z}{\partial \theta} + \breve{u}_z\frac{\partial \breve{u}_z}{\partial z} \\ = \frac{1}{Re}\left(\frac{\partial^2 U_z}{\partial r^2} + \frac{1}{r}\frac{\partial U_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 U_z}{\partial \theta^2} + \frac{\partial^2 U_z}{\partial z^2}\right. \\ \left. + \frac{\partial^2 \breve{u}_z}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_z}{\partial \theta^2} + \frac{\partial^2 \breve{u}_z}{\partial z^2}\right)\end{aligned} \tag{1.11d}$$

Two modifications are necessary and reduce these equations significantly. Terms of $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ refer to the leading order, base flow solution. Since we begin an analysis of this type knowing the base flow solution, the collection of these terms sum to exactly zero. Also, terms of $\mathcal{O}(\breve{m}^2)$ are second order and can be dismissed from the linear framework. The results are:

Continuity:

$$\frac{\partial \breve{u}_r}{\partial r} + \frac{\breve{u}_r}{r} + \frac{1}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\partial \breve{u}_z}{\partial z} = 0 \tag{1.12a}$$

Radial momentum:

$$\begin{aligned}\frac{\partial \breve{u}_r}{\partial t} + U_r\frac{\partial \breve{u}_r}{\partial r} + \breve{u}_r\frac{\partial U_r}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_r}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_r}{\partial \theta} - \frac{2U_\theta \breve{u}_\theta}{r} + U_z\frac{\partial \breve{u}_r}{\partial z} + \breve{u}_z\frac{\partial U_r}{\partial z} + \frac{\partial \breve{p}}{\partial r} \\ = \varepsilon\left(\frac{\partial^2 \breve{u}_r}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_r}{\partial r} - \frac{\breve{u}_r}{r^2} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_r}{\partial \theta^2} - \frac{2}{r^2}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\partial^2 \breve{u}_r}{\partial z^2}\right)\end{aligned} \tag{1.12b}$$

Tangential momentum:

$$\frac{\partial \breve{u}_\theta}{\partial t} + U_r \frac{\partial \breve{u}_\theta}{\partial r} + \breve{u}_r \frac{\partial U_\theta}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r \breve{u}_\theta}{r} + \frac{\breve{u}_r U_\theta}{r} + U_z \frac{\partial \breve{u}_\theta}{\partial z} + \breve{u}_z \frac{\partial U_\theta}{\partial z} + \frac{1}{r}\frac{\partial \breve{p}}{\partial \theta}$$
$$= \varepsilon \left(\frac{\partial^2 \breve{u}_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_\theta}{\partial r} - \frac{\breve{u}_\theta}{r^2} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_\theta}{\partial \theta^2} + \frac{2}{r^2}\frac{\partial \breve{u}_r}{\partial \theta} + \frac{\partial^2 \breve{u}_\theta}{\partial z^2} \right) \quad (1.12c)$$

Axial momentum:

$$\frac{\partial \breve{u}_z}{\partial t} + U_r \frac{\partial \breve{u}_z}{\partial r} + \breve{u}_r \frac{\partial U_z}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_z}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_z}{\partial \theta} + U_z \frac{\partial \breve{u}_z}{\partial z} + \breve{u}_z \frac{\partial U_z}{\partial z} + \frac{\partial \breve{p}}{\partial z}$$
$$= \varepsilon \left(\frac{\partial^2 \breve{u}_z}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_z}{\partial \theta^2} + \frac{\partial^2 \breve{u}_z}{\partial z^2} \right) \quad (1.12d)$$

where $\varepsilon = Re^{-1}$.

It is at this juncture that the path toward a solution diverges based on the method to be applied (OSE, LNP, PSE, BG). For complete derivations and further discussion of the LNP and BG stability equations see App. A.1 and App. A.2, respectively.

1.2.4 Parallel Flow Effects

In the study by Casalis and Vuillot [55], the validity of the normal mode assumption is discussed, specifically

$$m'(r, \theta, z, t) = m(r) \exp\left[i\left(q\theta + \alpha z - \omega t\right)\right] \quad (1.13)$$

The discussion also addresses its justification (or lack thereof) for two-dimensional base flows. In short, continuity imposes a dependence of the streamwise velocity on the streamwise coordinate through the nonzero transverse velocity. This is a *nonparallel* effect. Thus, the normal mode remains a valid approximation only for flows weakly dependent on the

streamwise coordinate, or

$$\frac{\partial M}{\partial z}, \frac{\partial M}{\partial \theta} \ll \frac{\partial M}{\partial r} \tag{1.14}$$

This is known as the *parallel flow assumption*. Setting the left-hand-side of this equality equal to exactly zero is the basis for deriving the Orr-Sommerfeld equation.

Casalis and Vuillot go further to propose a means of analyzing the degree in which the parallel flow assumption is violated. Since the degree of the nonparallel effects is related to the magnitude of the transverse velocity with respect to the streamwise coordinate, we can simply calculate their ratio to qualify their significance. We define the *Nonparallel Ratio* (NPR) as

$$\text{NPR} \equiv \frac{U_r}{U_z} \tag{1.15}$$

This calculation is performed for several base flows in Eqs. (1.16a–1.16c).

$$\text{NPR} = \left| \frac{-\sin(\pi r/2)}{\pi z/2 \cos(\pi r/2)} \right| \approx \frac{r}{z} + \frac{\pi^2 r^3}{12z} + \mathcal{O}(r^5) \qquad \text{Taylor Plane Flow} \tag{1.16a}$$

$$\text{NPR} = \left| \frac{-1/r \sin(\pi r^2/2)}{\pi z \cos(\pi r^2/2)} \right| \approx \frac{r}{2z} + \mathcal{O}(r^5) \qquad \text{Taylor-Culick Axisymmetric Flow} \tag{1.16b}$$

$$\text{NPR} = \left| \frac{-\kappa/r \sin(\pi r^2)}{2\pi\kappa z \cos(\pi r^2)} \right| \approx \frac{r}{2z} + \mathcal{O}(r^5) \qquad \text{Bidirectional Vortex Flow} \tag{1.16c}$$

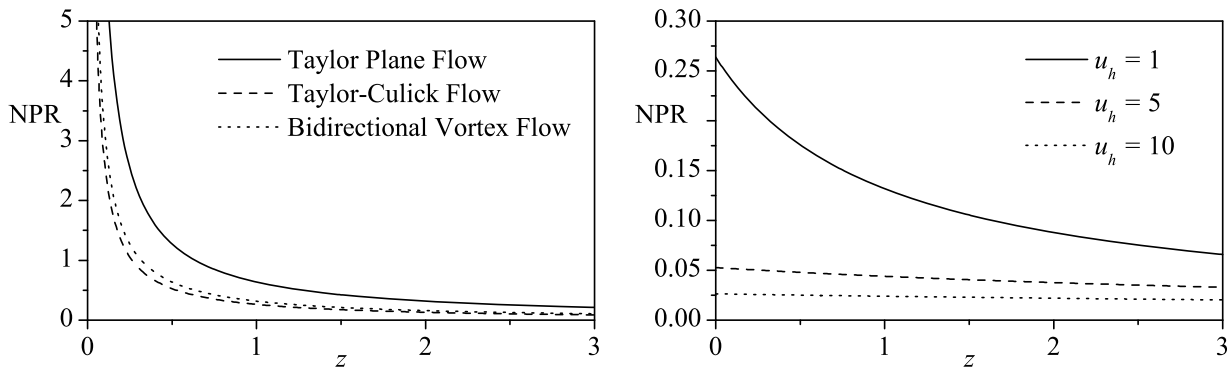
A $1/z$ effect can be clearly seen approaching the headwall. For long chambers $1/z$ diminishes significantly and therefore the nonparallel effects follow suit. However, for short chambers associated with the bidirectional vortex engine (BVE), we can safely assume nonparallel effects are indeed important. A graphical representation of these results is found in Fig. 1.3a. The radial coordinate acts as a scaling factor but must be taken away from the centerline or sidewall. At both $r = 0$ and $r = 1$, the radial velocity is strictly zero and the NPR will, likewise, be zero. The radial coordinate will maximize the NPR where the radial velocity is maximum but is inconsequential when compared to the effect of streamwise position.

For comparison purposes, we introduce another stability analysis performed by Abu-Irshaid, Majdalani, and Casalis [76]. Their study considers the Taylor-Culick base flow with uniform headwall injection. The particular configuration is appropriate for hybrid rockets with headwall injection. A similar calculation can be made to gage the importance of nonparallel effects, namely,

$$\mathrm{NPR} = \left| \frac{-1/r \sin(\pi r^2/2)}{\pi(z+u_h)\cos(\pi r^2/2)} \right| \approx \frac{r}{2(z+u_h)} + \mathcal{O}(r^5) \tag{1.17}$$

Right away we see that this result is slowly changing near the headwall and goes to zero as z gets large. Furthermore, the NPR remains bounded at the headwall because of a finite u_h and remains small over the entire length of the chamber. This result is plotted alongside the three previous flowfields for three values of headwall injection in Fig. 1.3b. We can conclude that this base flow is a viable candidate for a local nonparallel stability analysis where the previous three may only be for long chambers.

In seeking a biglobal stability analysis, we eliminate the deficiencies of the parallel flow assumption while sacrificing simplistic formulations and computational time. Solutions inherently consider both the transverse and streamwise coordinates so the end result is not sensitive to the streamwise position and is well suited for axisymmetric geometries.



a) Nonparallel ratio for three common base flows

b) Nonparallel ratio for Taylor-Culick flow with uniform headwall injection [77]

Figure 1.3: A qualitative comparison of the nonparallel effects of several base flows for $r = 0.5$.

Chapter 2

Vortex Flows

In this chapter, we will review some of the theoretical solutions used to describe swirl dominated flows in both unidirectional and bidirectional flow orientations. A brief historical time line discussing the evolution of vortex flow modeling as it developed from the external one-dimensional Rankine vortex to the most recent developments in confined multidirectional vortex flows will be presented. Many similarities are found between the models regardless of the context in which they were derived. The main thrust of this chapter, however, is the derivation of the multidirectional vortex flows that represent the basis for this hydrodynamic instability study.

2.1 An Introduction to Vortex Flows

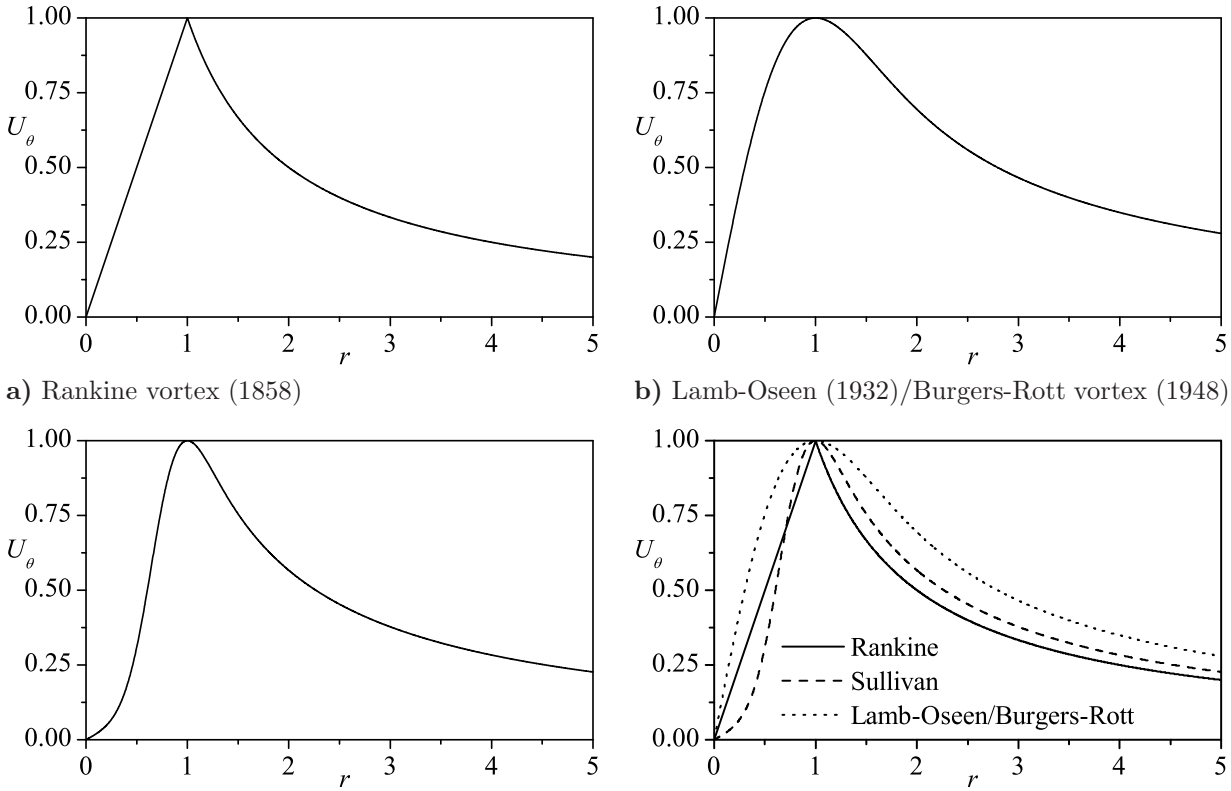
Coherent vortex structures in naturally occurring phenomena have peaked interest in their mathematical intricacy and potential practical applications for hundreds of years. Inducing vortex motion over non-rotating fluid flow is known to affect thermal and transport properties. Their favorable attributes are leveraged in industrial applications such as swirl combustors where vortex injection increases residence time and promotes efficient combustion and cyclonic separators where centrifugal forces cause heavier particles to migrate outwardly. Meteorological vortex flows are most notable for their destructive power.

Perhaps the first among analytic vortex models was introduced by Rankine in 1858 [2]. This combined vortex solution is a piecewise continuous function identifying two distinct vortex characteristics. When nondimensionalized by the characteristic boundary layer thickness, δ, it is expressed as:

$$U_\theta(r) = \frac{\bar{U}_\theta}{\left(\bar{U}_\theta\right)_{\max}} = \begin{cases} r, & 0 \leq r \leq 1 \\ r^{-1}, & r > 1 \end{cases} \qquad \text{where } r = \frac{\bar{r}}{\delta} \tag{2.1}$$

For $\bar{r} > \delta$ we observe what is called the *free vortex* tail. Here, the velocity is inversely proportional to $\bar{r}$. The inviscid free vortex is augmented by the viscous forced core in order to overcome the r^{-1} singularity at the centerline and produce a uniformly valid solution over the entire domain, $0 \leq r \leq \infty$. The region $\bar{r} \leq \delta$ is referred to as the *forced core vortex*. This domain is indicative of solid body rotation and is defined up to the maximum velocity, $\left(\bar{U}_\theta\right)_{\max}$. In this region, viscous forces overshadow inertial forces and give rise to a distinctly different solution from the outer, free vortex. Both pieces can be clearly seen in Fig. 2.1a.

The piecewise nature of the Rankine vortex is, simultaneously, its best and worst property. Its simplicity lends itself to quick comparisons with experimental data and, perhaps for this reason, it is still commonly used to curve fit data for swirl velocities in tornadoes [78] and hurricanes [79]. On the other hand, the one-dimensional model may be oversimplified for



a) Rankine vortex (1858)

b) Lamb-Oseen (1932)/Burgers-Rott vortex (1948)

c) Sullivan vortex (1959)

d) Nondimensional overlay of all four models

Figure 2.1: Normalized swirl velocity for several historic vortex models.

weakly swirl dominated flow regimes. Fortunately, the tangential velocity is at least an order of magnitude larger than either the radial or axial velocities in most swirling flows. Thus, it is no stretch to regard these other two vector components as secondary and dismiss them from the model. Lastly, the piecewise defined solution produces a cusp where the forced core and free vortex regions meet. This leads to an overshoot in both maximum velocity and forced core thickness. Experimenters can account for the overshoot by fitting the core and tail regions independently and extracting the core thickness and maximum velocity directly from experimental data.

The next vortex model of relevance is that of Oseen [4] and Lamb [3] circa 1932. The relationship between the two authors, if any, is unclear even though it has become the

convention to refer to this model as the Lamb-Oseen vortex. The corresponding formulation incorporates temporal decay by defining a time dependent characteristic boundary layer thickness. The model appears nondimensionally as

$$U_\theta\left(r\right) = \frac{1}{2\pi r}\left(1 - e^{-r^2}\right)$$

$$\text{where } r = \frac{\bar{r}}{\delta}; \quad U_\theta(r) = \frac{\bar{U}_\theta(\bar{r})}{\Gamma/\delta}; \quad \delta = 2\sqrt{\nu t}; \quad \text{and} \quad \Gamma = \iint_S (\nabla \times \boldsymbol{U}) \cdot \mathrm{d}\boldsymbol{S}$$

Without an energy source to sustain vortex motion, this model succumbs to the dissipative effects of shear stresses over time. These force the maximum velocity to diminish while the core thickness increases until the profile is damped to zero. Therefore, this solution is commonly used to model wing-tip vortices and other dissipative rotating flows [13, 80]. Figure 2.1b illustrates the tangential velocity associated with this model.

A closely related model to the Lamb-Oseen vortex is the 1948 solution developed by Burgers [5, 81] and Rott [6, 82, 83]. Both the Burgers-Rott and Lamb-Oseen vortex models are axisymmetric Gaussian vortex solutions of the incompressible Navier-Stokes equations. They can both be collapsed into the Gaussian form:

$$\bar{U}_\theta(r) = \Gamma G\left(\frac{\bar{r}}{\delta}\right); \quad G(x) = \frac{1}{4\pi} e^{-x^2/4} \tag{2.2}$$

Conversely, these two models differ in two ways: First, the Burger-Rott vortex is commonly accompanied by an axial and radial velocity component. Second, rather than time dependence, this model introduces a suction parameter, a, that accounts for any external strain field where the axial flow is generated by a pressure gradient [8]. The suction parameter, a, is originally named the "inflow-gradient" by Rott and is actually a constant gradient of velocity of flow coming into the vortex from the outer field [6]. It is approximated as a constant even in cases where it possesses weak axial dependence. For sink flow, it can

be explicitly defined as

$$a = \frac{Q}{2\pi h^3} \tag{2.3}$$

where Q is the sink strength and h is the distance to the infinite plane (the undisturbed free surface). The total nondimensional Burgers-Rott vortex is given as

$$U_\theta(r) = \frac{1}{2\pi r}\left(1 - e^{-r^2}\right);$$
$$U_r(r) = -r; \quad U_z(z) = 2z$$

$$\text{where} \quad r, z = \frac{r, z}{\delta}; \quad U_\theta(r) = \frac{\bar{U}_\theta(r)}{\Gamma/\delta}; \quad U_{r,z} = \frac{\bar{U}_{r,z}}{a\delta};$$
$$\delta = \sqrt{2\nu/a}; \quad \text{and} \quad \Gamma = \iint_S (\nabla \times \boldsymbol{U}) \cdot \mathrm{d}\boldsymbol{S}$$

Note that the nondimensional form of both the Lamb-Oseen and Burgers-Rott models are identical. Their plot is shared in Fig. 2.1b.

A closer consideration of the axial and radial velocity components identify both as linear, unbounded functions. This behavior implies that the vortex must be relatively small compared the strain field in which it is emersed. Often this caveat is circumvented by using only the tangential velocity as a purely one-dimensional model. One application in which all three vector components are retains is swirl modeling of tornadoes under a large thunderhead [84].

The first instance of a bidirectional vortex model dates back to 1959 when an exact solution of the Navier-Stokes equations in an unbounded domain was developed by Sullivan [14]. This vortex is characterized by a strong downdraft near the axis of symmetry and a weaker updraft caused by the outward turning of the core vortex after impinging upon a solid surface at $z = 0$. This type of motion was originally denoted as a *two-cell* vortex and is ideal for modeling tornadoes. We find this model to be slightly more difficult to apply since

it is resolved in terms of integral functions. We have

$$U_\theta(r) = \frac{1}{2\pi r}\frac{G\left(r^2\right)}{G\left(\infty\right)} \quad \text{with} \quad G(x) = \int_0^x e^{f(t)}\,\mathrm{d}t; \quad f(t) = -t + 3\int_0^t \left(1 - e^{-y}\right)\frac{\mathrm{d}y}{y};$$
$$U_r(r) = -r + \frac{3}{r}\left(1 - e^{-r^2}\right); \quad U_z(r,z) = 2z\left(1 - 3e^{-r^2}\right)$$

$$\text{where} \quad (r,z) = \frac{(\bar{r},\bar{z})}{\delta}; \quad U_\theta\left(r\right) = \frac{\bar{u}_\theta\left(\bar{r}\right)}{\Gamma/\delta}; \quad U_{r,z} = \frac{\bar{U}_{r,z}}{a\delta};$$
$$\delta = \sqrt{2\nu/a}; \quad \text{and} \quad \Gamma = \iint_S (\nabla \times \boldsymbol{U}) \cdot \mathrm{d}\boldsymbol{S}$$

Once again a refers to the suction strength and ν is the kinematic viscosity. Examination of the axial velocity shows the mantle location, or position along the radius where the axial velocity changes direction, to be at $r = \sqrt{\ln 3} \approx 1.04815$. Figure 2.1c shows the unique curvature of the forced core vortex as well as the familiar r^{-1} form of the free vortex region.

2.2 The Complex Lamellar Bidirectional Vortex

Although the bidirectional vortex is derived in the context of of propulsive applications these models are equally applicable to any bi- or multi-directional vortex flowfield that adheres to the basic geometry and assumptions introduced by their original developers. In the context of propulsion, the usage of vortex motion was first proposed to increase the burn-rate in hybrid rocket engines by Gloyer and coworkers [18]. This study laid the foundation for the development of the so-called Vortex Injection Hybrid Rocket Engine (VIHRE) and Vortex Combustion Cold-Wall Chamber (VCCWC) by Knuth *et al.* [19], Chiaverini *et al.* [20] and others.

2.2.1 The Invsicid Solution

The original inviscid solution was first presented in the conference paper by Vyas, Majdalani, and Chiaverini [25] that later appeared in modified form in AIAAJ [21]. It considers the case of a cylindrical tube of length L and radius a with a closed headwall and an exit port at the aft end whose dimensional radius is denoted by b. Purely tangential injectors are located at the base to induce swirl. This geometry is best visualized in Fig. 2.2. The axial velocity at the injectors is deemed small in comparison to the tangential and negligible in the mathematical model. Also, the distribution of tangential injection at the base is assumed to be uniform, thus indicative of a circular line source. This allows for the assumption of axisymmetry at the onset of injection. In practice, such an assumption can only be realized in a three-dimensional fluid after the flow has traversed a finite distance away from the aft end. Our governing equations are devised under the following set of principle conditions:

1. Steady
2. Inviscid
3. Axisymmetric
4. Incompressible
5. Rotational
6. Nonreactive/cold-flow
7. Axially independent swirl velocity

It is helpful to nondimensionalize all lengths by the radius and all velocities by the characteristic velocity (the tangential injection velocity). Other nondimensionalizations are devised on the basis of convenience and to eliminate extraneous dimensional constants in

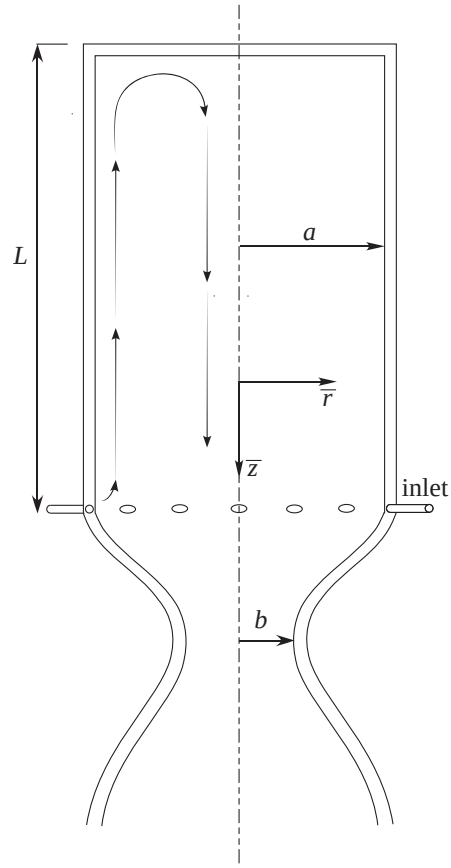


Figure 2.2: The bidirectional vortex geometry showing the coordinate system, characteristic dimensions, and general streamline profile.

the governing equations. As usual [21], we rely on the convention

$$
\begin{aligned}
z = \frac{\bar{z}}{a}; \quad r = \frac{\bar{r}}{a}; \quad \nabla = a\bar{\nabla}; \\
U_r = \frac{\bar{U}_r}{U}; \quad U_\theta = \frac{\bar{U}_\theta}{U}; \quad U_z = \frac{\bar{U}_z}{U}; \\
P = \frac{\bar{P}}{\rho U^2}; \quad Q_i = \frac{\bar{Q}_i}{U a^2}; \quad l = \frac{L}{a}
\end{aligned}
\tag{2.4}
$$

The derivation of this model begins by considering the nondimensional continuity and Euler equations with the above assumptions already fulfilled. Accordingly they are shown

to be

$$\nabla H - \boldsymbol{U} \times \boldsymbol{\Omega} = 0; \quad H = \tfrac{1}{2}\boldsymbol{U} \cdot \boldsymbol{U} + P \tag{2.5}$$

where H denotes the total fluid head. In component form

continuity:

$$\frac{1}{r}\frac{\partial\left(rU_r\right)}{\partial r} + \frac{\partial U_z}{\partial z} = 0 \tag{2.6}$$

r-momentum:

$$U_r\frac{\partial U_r}{\partial r} + U_z\frac{\partial U_r}{\partial z} - \frac{U_\theta^2}{r} = -\frac{\partial P}{\partial r} \tag{2.7}$$

θ-momentum:

$$U_r\frac{\partial U_\theta}{\partial r} + \frac{U_\theta U_r}{r} = 0 \quad \text{or} \quad \frac{1}{r}\frac{\partial(rU_\theta)}{\partial r} = 0 \tag{2.8}$$

z-momentum:

$$U_r\frac{\partial U_z}{\partial r} + U_z\frac{\partial U_z}{\partial z} = -\frac{\partial P}{\partial z} \tag{2.9}$$

At a glance, we see that Eq. (2.8) is decoupled from its radial and axial counterparts. This suggests that the tangential velocity is simply the potential vortex solution

$$U_\theta = r^{-1} \tag{2.10}$$

such that the boundary condition at the inlet, $U_\theta(1) = 1$, is satisfied.

The axial and radial solutions are devised by replacing the Euler equation with its equivalent vorticity streamfunction form. Recalling that the curl of a gradient is zero, we can first eliminate the stagnation pressure head by taking the curl of the momentum equation.

$$\nabla \times (\boldsymbol{U} \times \boldsymbol{\Omega}) = 0 \quad \longrightarrow \quad \frac{\partial\left(U_r\Omega_\theta\right)}{\partial r} + \frac{\partial\left(U_z\Omega_\theta\right)}{\partial z} = 0 \tag{2.11}$$

$$\boldsymbol{\Omega} = \nabla \times \boldsymbol{U} \quad \longrightarrow \quad \frac{\partial U_r}{\partial z} - \frac{\partial U_z}{\partial r} = \Omega_\theta \tag{2.12}$$

Equation (2.11) is known as the *vorticity transport equation* where Eq. (2.12) is simply the *vorticity equation.*. Next, it is necessary to define the Stokes streamfunction using

$$u_r = -\frac{1}{r}\frac{\partial \psi}{\partial z}; \quad u_z = \frac{1}{r}\frac{\partial \psi}{\partial r} \tag{2.13}$$

Even though our problem is fundamentally three-dimensional, we can utilize the streamfunction since the condition of axisymmetry along with an axially invariant tangential velocity leave the axial and radial velocities decoupled from their tangential counterpart.

By substituting the streamfunction into Eq. (2.11) and expanding where appropriate, we find

$$-\frac{\partial \psi}{\partial z}\frac{\partial}{\partial r}\left(\frac{\Omega_\theta}{r}\right) + \frac{\partial \psi}{\partial r}\frac{\partial}{\partial z}\left(\frac{\Omega_\theta}{r}\right) = 0 \tag{2.14}$$

The tangential component of vorticity is yet to be defined. In general, it is related to the tangential angular momentum such that it can be defined as $\Omega_\theta = rF[\psi(r,z)]$, where $F[\psi(r,z)]$ is some general function of the streamfunction. We are free to specify F such that we do not violate any physical conditions and recover a tractable analytic solution. There is an infinite number of possibilities but most are intractable or unphysical. To make headway, we define F to be a linear function, specifically $F = C_m^2\psi$. A similar analog was used by Culick in the context of gas motion in a solid rocket motor [85]. Making this substitution is the last step needed to convert the Euler equation into its equivalent vorticity streamfunction representation. We recover the linear, separable partial differential equation,

$$\frac{\partial^2 \psi}{\partial z^2} + \frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r}\frac{\partial \psi}{\partial r} + C_m^2 r^2 \psi = 0 \tag{2.15}$$

This equation is subject to the boundary conditions

$$z = 0; \quad U_z(r,0) = 0 \quad \text{or} \quad \frac{1}{r}\frac{\partial \psi\,(r,0)}{\partial r} = 0 \tag{2.16a}$$

$$r = 0; \quad U_r(0,z) = 0 \quad \text{or} \quad -\frac{1}{r}\frac{\partial \psi\,(0,z)}{\partial z} = 0 \tag{2.16b}$$

$$r = 1; \quad U_r(1,z) = 0 \quad \text{or} \quad -\frac{1}{r}\frac{\partial \psi\,(1,z)}{\partial z} = 0 \tag{2.16c}$$

$$\begin{aligned} z = l; \quad Q_i &= \int_0^\beta \int_0^{2\pi} \boldsymbol{U} \cdot \boldsymbol{n}\; r\,\mathrm{d}r\,\mathrm{d}\theta \\ &= 2\pi \int_0^\beta U_z(r,l) r\,\mathrm{d}r = 2\pi \int_0^\beta \frac{1}{r}\frac{\partial \psi(r,l)}{\partial r}\; r\,\mathrm{d}r \end{aligned} \tag{2.16d}$$

By separation of variables we find the general solution to be

$$\psi = (C_1 z + C_2)\left[C_3 \sin\left(\frac{C_m r^2}{2}\right) + C_4 \cos\left(\frac{C_m r^2}{2}\right)\right] \tag{2.17}$$

where the separation constant is specified to be zero for this solution.

Equation (2.16a) is a hardwall boundary condition that prevents axial flow at the headwall. This condition implies that $C_2 = 0$. We require this condition to be nonzero for hybrid rockets [26, 77, 86]. The condition enforcing axisymmetry by disallowing crossflow at the centerline is met by Eq. (2.16b); thus, we can infer that $C_4 = 0$. At present we have reduced the general solution to

$$\psi = \psi_0 z \sin\left(\frac{1}{2} C_m r^2\right); \quad \psi_0 = C_1 C_3 \tag{2.18}$$

Equation (2.16c) is the condition for nontranspiring sidewalls. It results in the eigenvalue problem $\sin(\frac{1}{2}C_m) = 0$. Solutions exist only when $C_m = 2\lambda_m$, where $\lambda_m = \pi, 2\pi, \ldots (m+1)\pi$. The fundamental solution is given by Vyas and Majdalani when $m = 0$ or $C_m = 2\pi$. Higher values of m lead to multi-directional flowfields [30, 33].

Lastly, we enforce conservation of mass through Eq. (2.16d). This results in the remaining constant to be $\psi_0 = Q_i/[2\pi l \sin(\pi\beta^2)]$ where β serves as the nondimensional exit port radius and also as the location of the mantle. The selection of the mantle radius to be the same as that of the exit port is necessary to create an unopposed exit flow [21]. By collecting the remaining parameters and utilizing the definition of the Stokes streamfunction, we have

$$\psi = \kappa z \sin\left(\pi r^2\right) \tag{2.19}$$

$$\boldsymbol{U} = -\frac{\kappa}{r}\sin\left(\pi r^2\right)\boldsymbol{e}_r + r^{-1}\boldsymbol{e}_\theta + 2\pi\kappa z\cos\left(\pi r^2\right)\boldsymbol{e}_z \tag{2.20}$$

The pressure is found by integrating the momentum equation. We find that

$$\Delta P = -\frac{1}{2r^2}\left\{1 + \frac{1}{2}\kappa^2\left[8\pi^2 r^2 z^2 + 1 - \cos\left(2\pi r^2\right)\right]\right\} \tag{2.21}$$

where $\kappa = Q_i/(2\pi l)$ is the inlet parameter.

The resulting velocity and pressure distributions are plotted in Fig. 2.3 and the streamlines in Fig. 2.4. We see clearly that the tangential velocity is singular at the centerline and, along with the axial velocity, fails to capture the small boundary layer forming along the sidewall. In the forthcoming section we will illustrate the procedure of boundary layer characterization via asymptotic methods. We will improve upon the inviscid profile by generating uniformly valid solutions that eliminate the tangential velocity's centerline singularity and encompass the effects of viscosity along the sidewall.

2.2.2 Viscous Corrections

Viscous corrections for the tangential velocity were suggested by Majdalani and Chiaverini [23] and for the axial and radial by Batterson and Majdalani [24]. These studies show how the boundary layer equations may be derived starting with the Navier-Stokes equations and dismissing terms that may be deemed of higher order in the viscous layer [87]. Then following

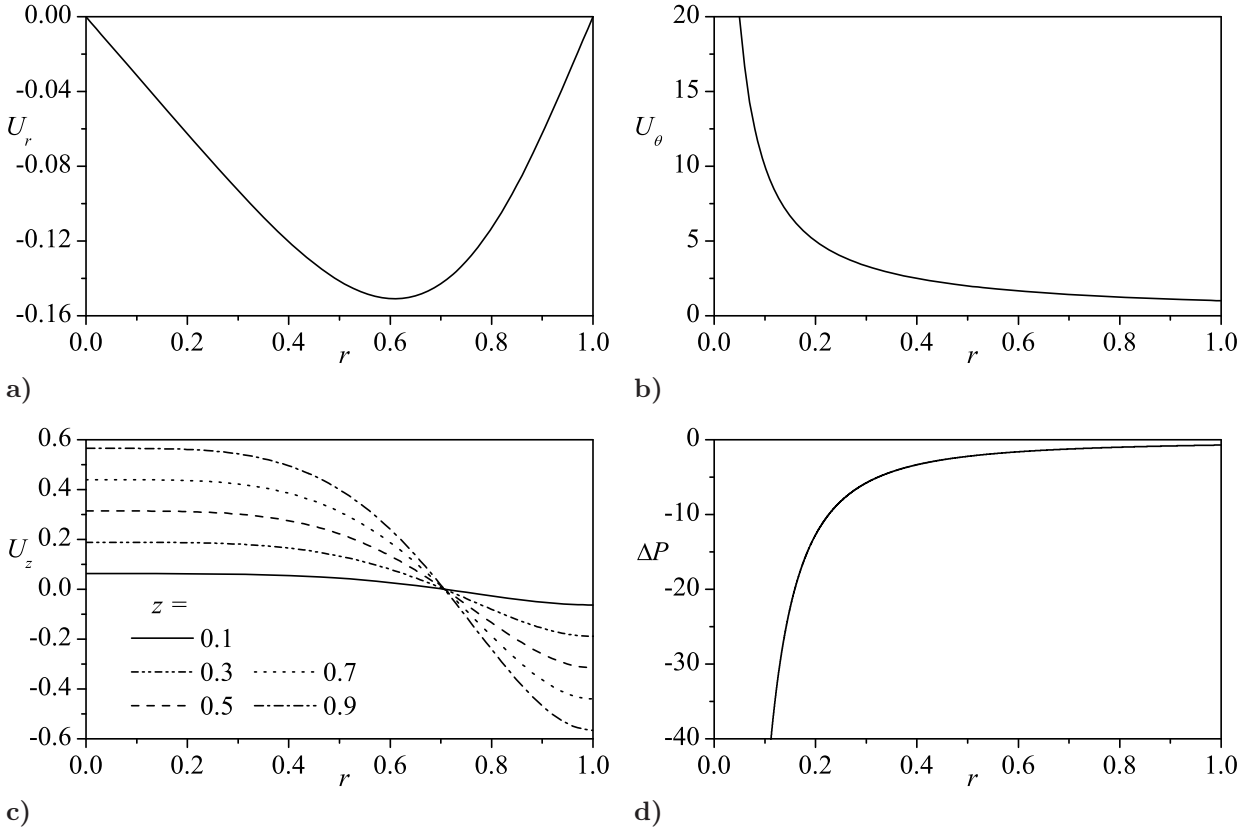


Figure 2.3: The inviscid complex-lamellar bidirectional vortex with $\kappa = 0.103$ and $z = 0.3$.

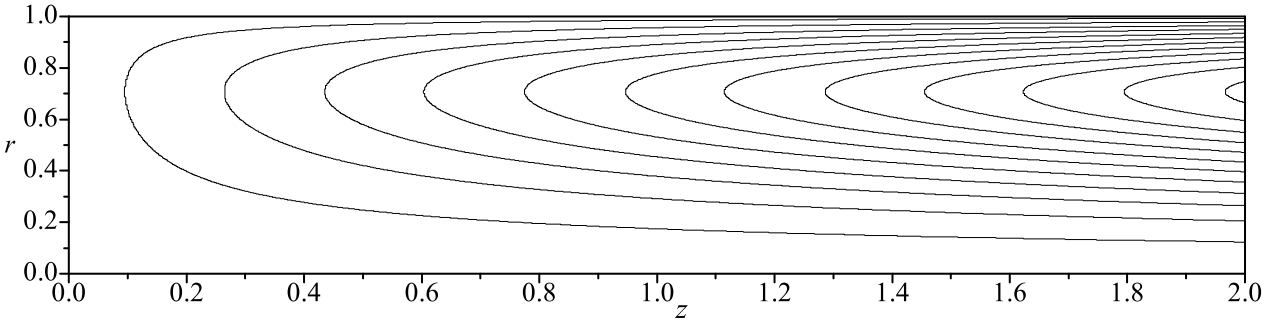


Figure 2.4: Streamlines of the complex-lamellar BV.

Schlichting [88], Tetervin [89] and others, the boundary layer equations are scaled. Finally, the new equations are linearized, solved asymptotically, and matched to the outer inviscid solution. In this section we will detail the analytic techniques used to compute the effects of viscosity and, in turn, generate the viscous complex-lamellar bidirectional vortex profile.

Tangential Core and Sidewall Corrections

Given the swirl dominated nature of the bidirectional vortex, it is appropriate to first address the effect of viscosity in the tangential velocity profile. The deficiencies of a purely inviscid solution can clearly be seen in Fig. 2.3b. First, the solution only generates the free vortex and possesses a singularity at the centerline. Second, it omits the presence of a shear layer along the sidewall. Both of these limitations can be addressed through asymptotic boundary layer analysis. The tangential momentum equation can be reduced by assuming that velocities and coordinate directions parallel to the primary flow direction are of order unity, while those perpendicular are order δ. Following convention, the dimensionless viscosity $\nu/(Ua)$ may be taken at $\mathcal{O}(\delta^2)$ [87]. Finally, terms of $\mathcal{O}(\delta)$ and above may be truncated, thus leaving

$$\frac{U_r}{r}\frac{\partial}{\partial r}\left(rU_\theta\right) + U_z\frac{\partial U_\theta}{\partial z} = \varepsilon\left\{\frac{\partial}{\partial r}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rU_\theta\right)\right]\right\}; \quad \varepsilon = \frac{\nu}{Ua} = Re^{-1} \tag{2.22}$$

This equation can be reduced further by noting that the axial invariance of the invisicid solution translates well to the boundary layer equations, and that derivatives with respect to z can be eliminated. From the original studies, the ensuing equation is now subject to the dependent-variable transformation, $U_\theta = \xi_\theta(r)/r$. It can be stated as the transformed ODE

$$\varepsilon\left(\frac{\mathrm{d}^2\xi_\theta}{\mathrm{d}r^2} - \frac{1}{r}\frac{\mathrm{d}\xi_\theta}{\mathrm{d}r}\right) - U_r\frac{\mathrm{d}\xi_\theta}{\mathrm{d}r} = 0 \tag{2.23}$$

The radial velocity is not subject to viscous effects along the centerline since this region is not a site of increased shear for neither the axial nor radial velocity components.

Furthermore, previous studies have shown an inconsequential effect of viscosity on the radial velocity profile along the sidewall. This lends itself to being a reasonable first order approximation to the convective coefficient in the above equation in the core region and near the sidewall. By inserting the inviscid radial velocity directly into the governing equation, we effectively linearize the problem. We have

$$\varepsilon \frac{\mathrm{d}}{\mathrm{d}r}\left(r^{-1}\frac{\mathrm{d}\xi_\theta}{\mathrm{d}r}\right) + \kappa r^{-2}\sin\left(\pi r^2\right)\frac{\mathrm{d}\xi_\theta}{\mathrm{d}r} = 0 \tag{2.24}$$

Next, the original authors employed the variable transformation $\eta = \pi r^2$ to reduce the argument of the sine function. This last transformation converts the tangential boundary layer equation into its final form wherein it is valid for both the core and sidewall upon proper scaling. This equation is

$$\varepsilon \frac{\mathrm{d}^2\xi_\theta}{\mathrm{d}\eta^2} + \frac{\kappa \sin\eta}{2\eta}\frac{\mathrm{d}\xi_\theta}{\mathrm{d}\eta} = 0 \tag{2.25}$$

The linear scale $s = \eta/\delta$ is applied to Eq. (2.25) to distinguish the viscous core from the sidewall region. Once again δ represents the characteristic boundary layer thickness. This layer scales the equation such that $0 \leq \eta \leq \delta$ now corresponds to $0 \leq s \leq 1$. The new equation becomes

$$\frac{\varepsilon}{\kappa\delta}\frac{\mathrm{d}^2\xi_\theta^{(i)}}{\mathrm{d}s^2} + \frac{\sin\left(\delta s\right)}{2\delta^2 s}\frac{\mathrm{d}\xi_\theta^{(i)}}{\mathrm{d}s} = 0 \tag{2.26}$$

In order for the diffusive and convective terms to balance, the defining parameters controlling their respective magnitudes must balance accordingly. Here the three small parameters, ε, δ, and κ, must be considered. A Taylor series expansion of the sine is employed to determine the proper order of the convective term. To first order, we identify $\sin(\delta s) \approx \delta s$. Thus we can represent our equation by

$$\frac{\mathrm{d}^2\xi_\theta^{(i)}}{\mathrm{d}s^2} + \frac{1}{2}\frac{\mathrm{d}\xi_\theta^{(i)}}{\mathrm{d}s} = 0 \tag{2.27}$$

where the boundary layer thickness is defined by the distinguished limit as $\delta \sim \varepsilon/\kappa$. Fortuitously, by only retaining the leading-order expansion of the convective coefficient, we incur a maximum local error of 2-10% for realistic values of ε and κ when compared to the numerical solution.

The associated boundary conditions can be written as

$$\begin{cases} \xi_\theta^{(i)}(0) = 0 \\ \lim\limits_{s\to\infty} \xi_\theta^{(i)} = \lim\limits_{r\to 0} \xi_\theta^{(o)} = 1 \end{cases} \tag{2.28}$$

where $\xi_\theta^{(o)} = r u_\theta^{(o)}$. The conditions in Eq. (2.28) act to remove the singularity at the centerline and to asymptotically approach the inviscid profile in the outer domain in accordance with Prandtl's matching principle [90].

The general solution to Eq. (2.27) is simply

$$\xi_\theta^{(i)} = A + Be^{-s/2} \tag{2.29}$$

Applying the first boundary condition yields

$$\xi_\theta^{(i)} = A\left(1 - e^{-s/2}\right) \tag{2.30}$$

Accordingly, the second boundary condition divulges that $A = 1$ and

$$\xi_\theta^{(i)} = 1 - e^{-s/2} \tag{2.31}$$

The sidewall corrections follow a similar procedure but under the sidewall scaling transformation $s = (\pi - \eta)/\delta$.

This time Eq. (2.25) becomes

$$\frac{\varepsilon}{\delta^2}\frac{\mathrm{d}^2\xi_\theta^{(w)}}{\mathrm{d}s^2} - \frac{\kappa\sin\left(\pi-\delta s\right)}{2\delta\left(\pi-\delta s\right)}\frac{\mathrm{d}\xi_\theta^{(w)}}{\mathrm{d}s} = 0 \tag{2.32}$$

Again the coefficient in the convective term is expanded via Taylor series. Using the first two terms of the expansion of $\sin(x)$, we find as a brute approximation:

$$\begin{aligned}
-\frac{\kappa\sin\left(\pi-s\delta\right)}{2\delta\left(\pi-s\delta\right)} &\sim -\frac{\kappa}{2\delta\left(\pi-s\delta\right)}\left[\pi - s\delta - \frac{\left(\pi-s\delta\right)^3}{6}\right] \\
&= -\frac{\kappa}{2\delta}\left[1-\frac{\left(\pi-s\delta\right)^2}{6}\right] \sim -\frac{\kappa}{2\delta}\left(1-\frac{\pi^2}{6}\right) = \frac{\kappa\alpha}{2\delta}
\end{aligned}$$

This expansion identifies the proper scaling and significantly reduces the governing equation to

$$\frac{\varepsilon}{\delta^2}\frac{\mathrm{d}^2\xi_\theta^{(w)}}{\mathrm{d}s^2} + \frac{\kappa\alpha}{2\delta}\frac{\mathrm{d}\xi_\theta^{(w)}}{\mathrm{d}s} = 0 \tag{2.33}$$

As before the distinguished limit is $\delta \sim \varepsilon/\kappa$ and the governing equation becomes

$$\frac{\mathrm{d}^2\xi_\theta^{(w)}}{\mathrm{d}s^2} + \frac{\alpha}{2}\frac{\mathrm{d}\xi_\theta^{(w)}}{\mathrm{d}s} = 0 \tag{2.34}$$

Our new boundary conditions are

$$\begin{cases} \xi_\theta^{(w)}\left(0\right) = 0 \\ \lim\limits_{s\to\infty}\xi_\theta^{(w)}\left(s\right) = \lim\limits_{r\to 1}\xi_\theta^{(o)}\left(r\right) = 1 \end{cases} \tag{2.35}$$

It can be shown that the complete solution to this set is

$$\xi_\theta^{(w)} = 1 - e^{-\alpha s/2} \tag{2.36}$$

A general, uniformly valid solution over the entire domain can be constructed according to Erdélyi's method of composite expansions [91]. This method states that a composite expansion can be constructed by summing the scaled solutions and subtracting the common part. Here it translates into

$$\xi_\theta^{(c)} = \xi_\theta^{(o)} + \xi_\theta^{(i)} + \xi_\theta^{(w)} - \lim_{s\to\infty} \xi_\theta^{(i)}(s) - \lim_{s\to\infty} \xi_\theta^{(w)}(s) \tag{2.37}$$

and in terms of original variables,

$$U_\theta^{(c)} = r^{-1}\left[1 - e^{-\frac{V}{4}r^2} - e^{-\frac{V}{4}\alpha\left(1-r^2\right)}\right]; \quad V = \frac{2\pi\kappa}{\varepsilon} \tag{2.38}$$

where V is denoted as the *vortex Reynolds number*.*

Axial and Radial Corrections

The characterization of the axial sidewall boundary layer follows that of the tangential. In fact, the analysis must conform identically. To put this statement in context: Consider the character of the resultant boundary layer. The profile formed from the vector sum of all three velocity components will most closely mimic that of the tangential velocity alone. Thus, the axial and radial velocities must not modify the boundary layer physically or mathematically from the form predicted in the tangential velocity. From a mathematical perspective, all three boundary layer equations balance diffusion with equivalent radial convection terms. Hence, any deviation in analysis would lead to a nonconforming mathematical solution and must be avoided. For these reasons, the assumptions used to reduce the tangential momentum equation to the solution of the boundary layer equation must be applied judiciously to the other two vector directions. It becomes apparent that the governing equation for the axial

*First defined by Majdalani (see [23])

boundary layer must be

$$\varepsilon r^{-1}\frac{\partial}{\partial r}\left(r\frac{\partial U_z}{\partial r}\right)+\kappa r^{-1}\sin\left(\pi r^2\right)\frac{\partial U_z}{\partial r}=-4\pi^2\kappa^2 z \tag{2.39}$$

where the inviscid radial velocity has already been substituted to linearize the governing equations. The right-hand-side refers to the inviscid pressure profile. Retention of the inviscid pressure profile in the viscous boundary layer equation is inconsequential since it is shown to appear at significantly higher order. The variable transformation $\eta = \pi r^2$ is applied once again to get

$$\varepsilon\left(\frac{\partial^2 U_z}{\partial \eta^2}+\frac{1}{\eta}\frac{\partial U_z}{\partial \eta}\right)+\frac{\kappa}{2\eta}\sin\left(\eta\right)\frac{\partial U_z}{\partial \eta}=-\frac{\pi\kappa^2 z}{\eta} \tag{2.40}$$

The sidewall scaling transformation, $s = (\pi - \eta)/\delta$, is applied to shift the domain to the wall region. We arrive at

$$\frac{\varepsilon}{\delta^2}\left(\frac{\partial^2 U_z^{(w)}}{\partial s^2}-\frac{\delta}{\pi - s\delta}\frac{\partial U_z^{(w)}}{\partial s}\right)-\frac{\kappa}{2\delta\left(\pi - s\delta\right)}\sin\left(\pi - s\delta\right)\frac{\partial U_z}{\partial s}=-\frac{\pi\kappa^2 z}{\pi - s\delta} \tag{2.41}$$

The convective coefficient is expanded in the same fashion as in the tangential velocity. Upon substitution the equation is reduced to

$$\frac{\partial^2 u_z}{\partial s^2}+\frac{\alpha}{2}\frac{\partial u_z}{\partial s}=0 \tag{2.42}$$

By now the necessity for a dependent variable transformation has been well documented [8, 24, 28, 29, 33]. In short, for Prandtl's matching principle to be applicable, the limit of the outer solution in the inner domain must be bounded and nonzero. This is true for both the axial and tangential velocity profiles. It is, however, not true for the radial profile. For the sake of consistency in the final solution, we are required to transform all three accordingly. Seeking solutions for the tangential and axial velocities without this final

variable transformation result in asymptotically equivalent solutions for large V ($V > 100$). In general, any transformation that results in a constant limit of the outer solution in the inner domain is viable; however, it is simpler to define said transformation to resemble the inviscid solution. Here we set $U_z^{(w)} = \xi_z^{(w)}(s)\,\pi z \cos(\pi - 2\pi s V^{-1})$ which is simply the functional shape of the inviscid solution scaled to the wall domain. Careful application of the chain rule shows that the derivatives are transformed into

$$\begin{aligned}
\frac{\partial U_z^{(w)}}{\partial s} &= -\pi z \cos\left(2\pi \frac{s}{V}\right) \frac{\mathrm{d}\xi_z^{(w)}}{\mathrm{d}s} + 2\pi^2 \frac{z}{V} \sin\left(2\pi \frac{s}{V}\right) \xi_z^{(w)} \\
&\simeq -\pi z \cos\left(2\pi \frac{s}{V}\right) \frac{\mathrm{d}\xi_z^{(w)}}{\mathrm{d}s} \\
\frac{\partial^2 U_z^{(w)}}{\partial s^2} &= -\pi z \cos\left(2\pi \frac{s}{V}\right) \frac{\mathrm{d}^2\xi_z^{(w)}}{\mathrm{d}s^2} + 4\pi^2 \frac{z}{V} \sin\left(2\pi \frac{s}{V}\right) \frac{\mathrm{d}\xi_z^{(w)}}{\mathrm{d}s} + 4\pi^3 \frac{z}{V^2} \cos\left(2\pi \frac{s}{V}\right) \xi_z^{(w)} \\
&\simeq -\pi z \cos\left(2\pi \frac{s}{V}\right) \frac{\mathrm{d}^2\xi_z^{(w)}}{\mathrm{d}s^2}
\end{aligned}$$

At leading order, this equation is now identical to Eq. (2.34). As such we have

$$\frac{\mathrm{d}^2\xi_z^{(w)}}{\mathrm{d}s^2} + \frac{\alpha}{2} \frac{\mathrm{d}\xi_z^{(w)}}{\mathrm{d}s} = 0 \tag{2.43}$$

with the boundary conditions

$$\begin{cases} \xi_z^{(w)}(0) = 0 \\ \lim\limits_{s\to\infty} \xi_z^{(w)}(s) = \xi_z^{(o)} = 2\kappa \end{cases} \tag{2.44}$$

From this point onward, the construction of the composite solution is straight-forward. After some work, we show that [24]

$$U_z^{(c)}(r,z) = 2\pi\kappa z \cos\left(\pi r^2\right) \left[1 - e^{-\frac{V}{4}\alpha(1-r^2)}\right] \tag{2.45}$$

The author has shown how the same procedure can be applied to the radial momentum equation [24] to recover

$$U_r^{(c)}(r) = -\kappa r^{-1} \sin\left(\pi r^2\right)\left[1 - e^{-\frac{V}{4}\alpha(1-r^2)}\right] \tag{2.46}$$

The pressure is deduced by integrating the Euler equations with the newly formed composite velocity profiles. It gives

$$\begin{aligned}\Delta P = -\frac{1}{2r^2}\Bigg\{&\left(1 - e^{-\frac{1}{4}Vr^2}\right)^2 + \left[1 - e^{-\frac{1}{4}\alpha V(1-r^2)}\right]^2 - 1 + 2e^{\frac{1}{4}V\left[\alpha+(1-\alpha)r^2\right]}\Bigg\} \\ &- \frac{1}{4}V\Bigg\{\mathrm{Ei}\left(-\tfrac{1}{2}Vr^2\right) - \mathrm{Ei}\left(-\tfrac{1}{4}Vr^2\right) + \alpha e^{-\frac{1}{4}\alpha V}\left[\mathrm{Ei}\left(\tfrac{1}{4}\alpha Vr^2\right) - \mathrm{Ei}\left(\tfrac{1}{4}\alpha V\right)\right] \\ &\qquad -\alpha e^{-\frac{1}{2}\alpha V}\left[\mathrm{Ei}\left(\tfrac{1}{2}\alpha Vr^2\right) - \mathrm{Ei}\left(\tfrac{1}{2}\alpha V\right)\right]\Bigg\}\end{aligned} \tag{2.47}$$

Figure 2.5 shows the uniformly valid composite solutions to the complex-lamellar bidirectional vortex. The centerline singularity in the tangential velocity is clearly absolved with the formation of a forced core vortex. The sidewall boundary layer is also accounted for in all three solutions. The effect of our viscous parameter, the so-called vortex Reynolds number, is displayed clearly. In fact, as $V \to \infty$, the inviscid solution is recovered exactly. A complete study of the viscous complex-lamellar bidirectional vortex is presented by Batterson and Majdalani [24].

2.3 The Beltramian Bidirectional Vortex

Recent work by Majdalani [26] spurred the advent of a Beltramian class of solutions to the bidirectional vortex. These two new solutions are characterized by a zero Lamb vector $\boldsymbol{\omega} \times \boldsymbol{u} = 0$, wherein their vorticity and velocity remain directly proportional. Unfortunately, the inviscid nature of these models is their bane. Similar viscous treatments to the one

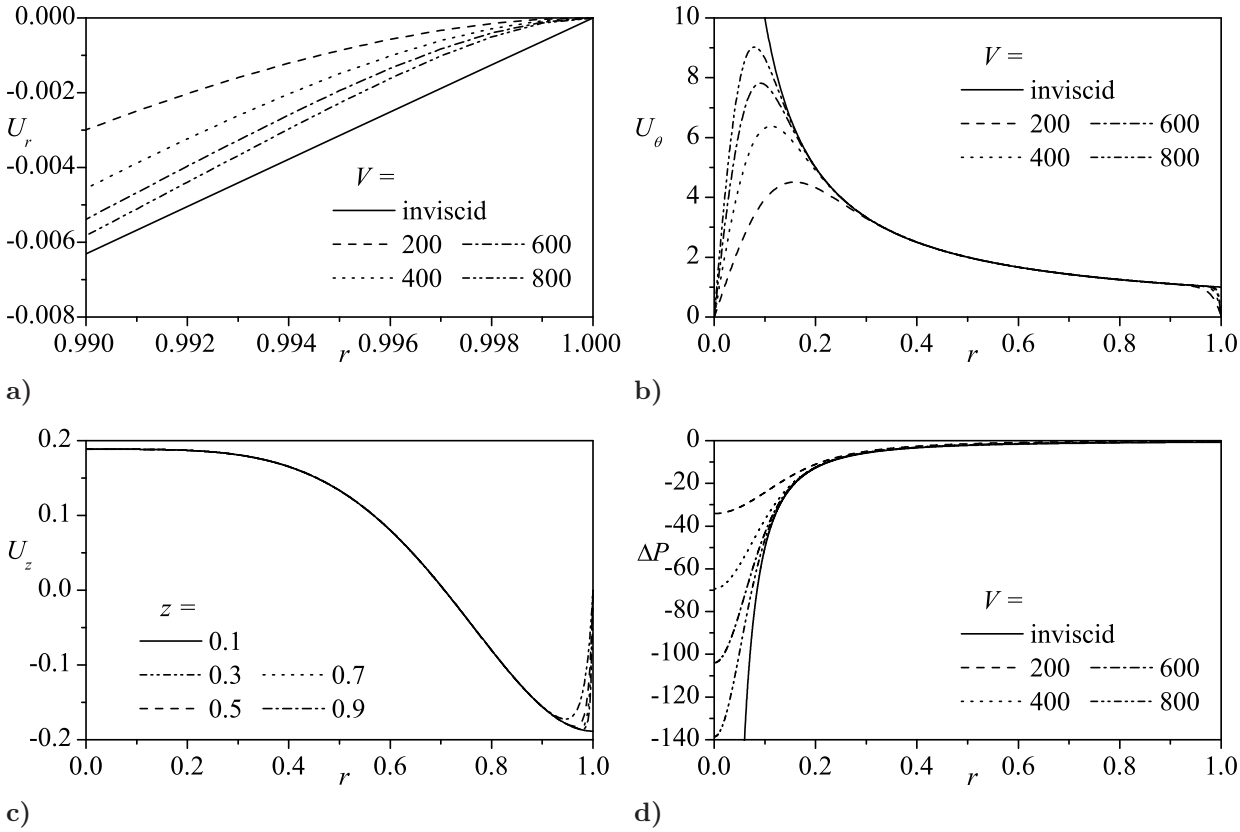


Figure 2.5: The viscous complex-lamellar bidirectional vortex with $\kappa = 0.103$ and $z = 0.3$.

presented above are required to properly eliminate the centerline singularity of the tangential velocity and to characterize the sidewall boundary layers in all three vector directions.

Rather than converting the Euler equations into their vorticity streamfunction representation, the Beltramian models are derived from a new direction. Here we seek to solve the Bragg-Hawthorne equation [92] instead of the Helmholtz type used by Vyas and Majdalani [21]. In reality these equations are very similar but retain enough differences to produce significantly different models. To begin we consider Eqs. (2.7–2.9) with the addition of the axisymmetric vorticity equation given by

$$\boldsymbol{\omega} = -\frac{\partial U_\theta}{\partial z}\boldsymbol{e}_r + \left(\frac{\partial U_r}{\partial z} - \frac{\partial U_z}{\partial r}\right)\boldsymbol{e}_\theta + \frac{1}{r}\frac{\partial (rU_\theta)}{\partial r}\boldsymbol{e}_z \tag{2.48}$$

As before, the $\theta-$momentum equation can be integrated automatically. This time, however, we choose to leave it in a more general form, namely,

$$rU_\theta = B(\psi) \tag{2.49}$$

where $B(\psi)$ stands for the tangential angular momentum defined in terms of the stream-function. This result can be substituted into Eq. (2.48) to give

$$\omega_r = -\frac{\partial u_\theta}{\partial z} = -\frac{1}{r}\frac{\partial (ru_\theta)}{\partial z} = -\frac{1}{r}\frac{\partial B}{\partial z} = -\frac{1}{r}\frac{\mathrm{d}B}{\mathrm{d}\psi}\frac{\partial \psi}{\partial z} \tag{2.50}$$

The tangential vorticity may be retrieved from the axial component of Eq. (2.5). We find

$$\frac{\partial H}{\partial z} + u_\theta \omega_r - u_r \omega_\theta = \frac{\mathrm{d}H}{\mathrm{d}\psi}\frac{\partial \psi}{\partial z} + u_\theta \omega_r - u_r \omega_\theta \tag{2.51}$$

Upon substitution of Eq. (2.13), Eq. (2.49), and Eq. (2.50) into Eq. (2.51), ω_θ can be determined. We find

$$\frac{\mathrm{d}H}{\mathrm{d}\psi}\frac{\partial \psi}{\partial z} - \frac{B}{r^2}\frac{\mathrm{d}B}{\mathrm{d}\psi}\frac{\partial \psi}{\partial z} + \frac{1}{r}\frac{\partial \psi}{\partial z}\omega_\theta = 0 \quad \text{or} \quad \frac{\omega_\theta}{r} = -\frac{\mathrm{d}H}{\mathrm{d}\psi} + \frac{B}{r^2}\frac{\mathrm{d}B}{\mathrm{d}\psi} \tag{2.52}$$

Finally, ω_θ can be substituted into Eq. (2.48) to arrive at the cylindrical Bragg-Hawthorne equation:

$$\frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r}\frac{\partial \psi}{\partial r} + \frac{\partial^2 \psi}{\partial z^2} = r^2 \frac{\mathrm{d}H}{\mathrm{d}\psi} - B\frac{\mathrm{d}B}{\mathrm{d}\psi} \tag{2.53}$$

The swirl velocity is indirectly included in the Bragg-Hawthorne equation in the tangential angular momentum term. It is at the discretion of the researcher to specify its form in accordance with the physical conditions. An infinite number of possibilities exist but, presently, most lead to intractable analytic solutions. Two simple solutions have already

been explored. They are

$$\begin{cases} B\dfrac{\mathrm{d}B}{\mathrm{d}\psi} = 0; & B = B_0 = 1 & \text{Vyas and Majdalani [21]} \\ B\dfrac{\mathrm{d}B}{\mathrm{d}\psi} = \text{const}; & B = \sqrt{B_0\psi + B_1} & \text{Bloor and Ingham [16]} \end{cases} \tag{2.54}$$

Here we see the close relationship between the Bragg-Hawthorne equation and the vorticity streamfunction equation used to obtain the complex-lamellar model. New analytic solutions can be recovered if we allow this term to be a linear function of ψ such that

$$B\frac{\mathrm{d}B}{\mathrm{d}\psi} = f(\psi) = C_m^2\psi; \quad B = \sqrt{C_m^2\psi^2 + B_1} \tag{2.55}$$

The constant C_m^2 is defined in this fashion to reduce clutter in later calculations. Moreover, the Bragg-Hawthorne equation may be simplified by recalling that total enthalpy is invariant along individual streamlines, $\mathrm{d}H/\,\mathrm{d}\psi = 0$. We are left with the linear, separable, partial differential equation,

$$\frac{\partial^2\psi}{\partial r^2} - \frac{1}{r}\frac{\partial\psi}{\partial r} + \frac{\partial^2\psi}{\partial z^2} + C_m^2\psi = 0 \tag{2.56}$$

As is customary, we consider the decomposition $\psi(r,z) = f(r)g(z)$ and substitute the result into Eq. (2.56). We are left with

$$-\frac{\ddot{g}}{g} = \frac{1}{f}\left(f'' - \frac{1}{r}f' + C_m^2 f\right) = \begin{cases} 0 \\ +\upsilon^2 \\ -\upsilon^2 \end{cases} \tag{2.57}$$

The first eigenvalue constitutes linear axial dependence; the second, harmonic axial dependence; and the third, a seemingly unphysical result.

2.3.1 The Linear Beltramian Solution

The linear Beltramian solution comes from the choice of $\upsilon = 0$. This choice results in the general solution for the streamfunction:

$$\psi = r\left(C_1 z + C_2\right)\left[C_3 J_1\left(C_m r\right) + C_4 Y_1\left(C_m r\right)\right] \tag{2.58}$$

The boundary conditions are equivalent to those for the complex-lamellar solution. They are found in Eq. (2.16). Equation (2.16a) considers a hardwall boundary condition that prevents injection at the headwall. For zero axial injection at the headwall, we require $C_2 = 0$. Equation (2.16b) may be used to enforce axisymmetry by disallowing crossflow along the centerline. However, in conjunction with Eq. (2.16a), this condition serves to enforce a bounded solution in the case of a Bessel type function. Hence, we must have $C_4 = 0$. The remaining expression becomes

$$\psi = \psi_0 z r J_1\left(C_m r\right); \quad \psi_0 = C_3 C_1 \tag{2.59}$$

Now, Eq. (2.16c) may be used to impose zero transpiration along the sidewalls. Like its predecessor, this condition gives rise to an eigenvalue equation of the form $J_1(C_m) = 0$. Its eigenvalues correspond to the roots of the Bessel function of the first kind, $C_m = \lambda_m$, where $\lambda_m = \{3.83171, 7.01559, 10.1735, 13.3237, ...\}$. Since this section is concerned with simple bidirectional motions, only the case of $\lambda_m = \lambda_0 = 3.83171$ will be considered here.

Finally, Eq. (2.16d) may be used to enforce mass conservation by requiring the injected flow to evacuate the chamber through the exit port. This fixes the last constant at $\psi_0 = Q_i/[2\pi l \beta J_1(\lambda_0 \beta)]$. Thus, we have

$$\psi = \kappa z r \frac{J_1\left(\lambda_0 r\right)}{\beta J_1\left(\lambda_0 \beta\right)} \tag{2.60}$$

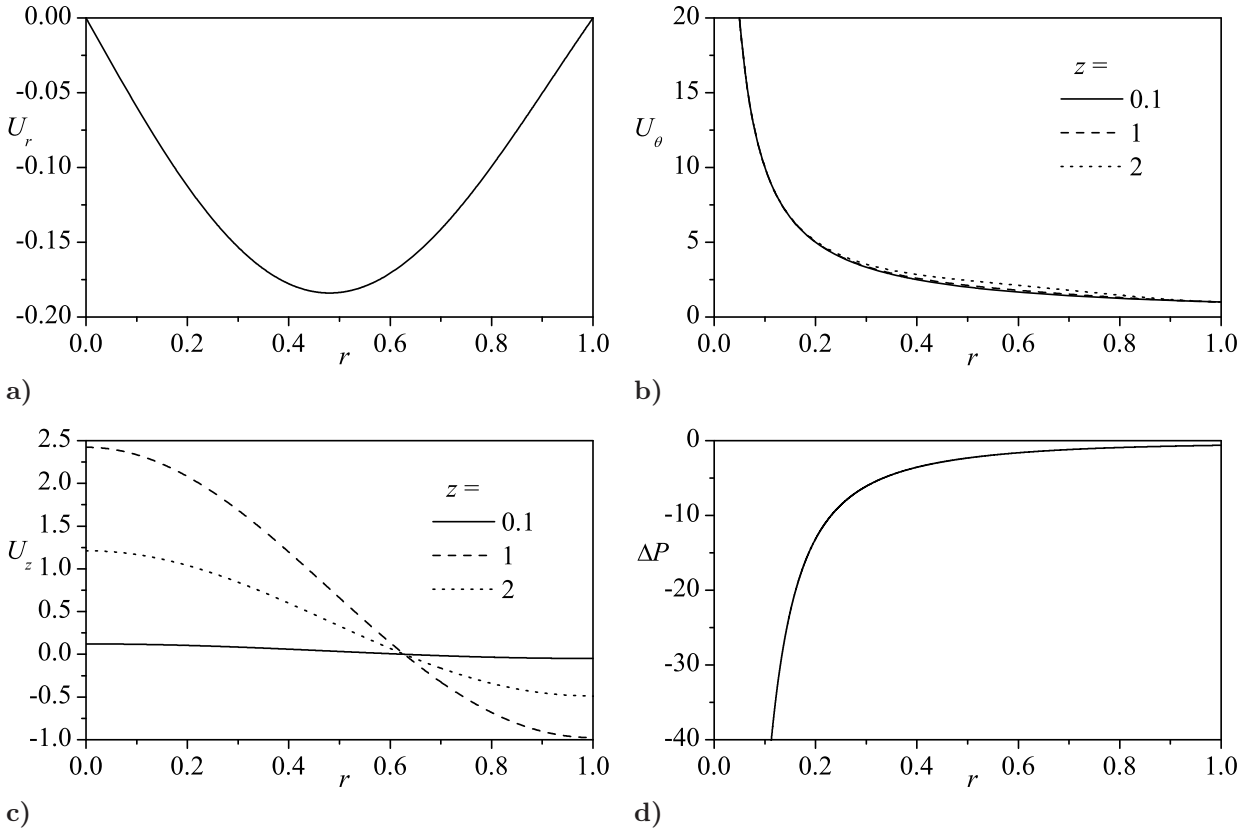


Figure 2.6: The inviscid linear Beltramian bidirectional vortex with $\kappa = 0.103$ and $z = 0.3$.

where $\kappa = Q_i/(2\pi l)$.

Now that the streamfunction is known, the axial and radial velocities can be calculated directly through Eq. (2.13). Also, along with the boundary condition $U_\theta(1, l) = 1$, the tangential velocity can be rectified via Eq. (2.49). The complete inviscid profile becomes

$$\boldsymbol{U} = -\kappa \frac{J_1(\lambda_0 r)}{\beta J_1(\lambda_0 \beta)} \boldsymbol{e}_r + r^{-1} \sqrt{1 + \frac{\lambda_0^2 \kappa^2 r^2 z^2 J_1^2(\lambda_0 r)}{\beta^2 J_1^2(\lambda_0 \beta)}} \boldsymbol{e}_\theta + \lambda_0 \kappa z \frac{J_0(\lambda_0 r)}{\beta J_1(\lambda_0 \beta)} \boldsymbol{e}_z \tag{2.61}$$

$$\Delta P = -\frac{1}{2r^2} - \frac{\kappa^2}{2\beta^2 J_1^2(\lambda_0 \beta)} \left\{ J_1^2(\lambda_0 r) + \lambda_0^2 z^2 \left[J_0^2(\lambda_0 r) + J_1^2(\lambda_0 r) \right] \right\} \tag{2.62}$$

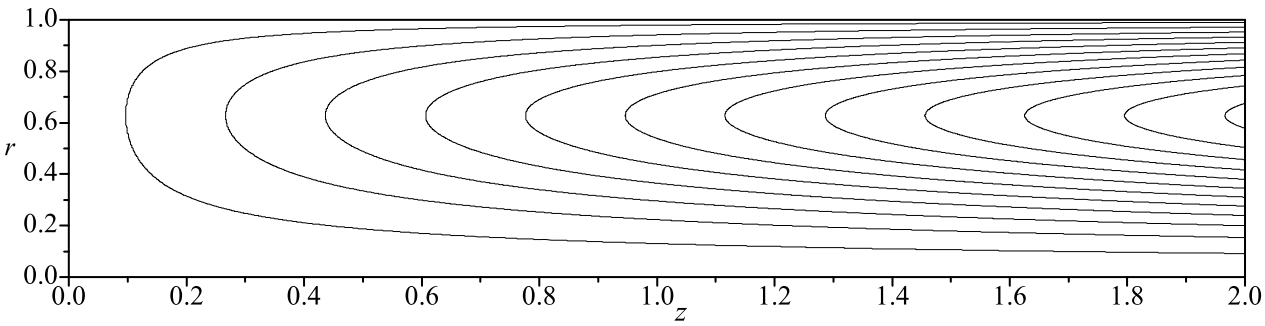


Figure 2.7: Streamlines of the linear Beltramian BV.

2.3.2 Viscous Corrections

Once again in Fig. 2.6 one may observe the same inviscid deficiencies along the centerline and sidewall that were witnessed with the complex-lamellar solution. To overcome these problems we will detail the procedure used by Batterson and Majdalani [28]. Being similar to the previous section and, for the sake of brevity, the analysis will be glossed over quickly while referring the reader to the original paper or the previous section for detail. The streamlines in Fig. 2.7 are similar to the those of the complex-lamellar model shown in Fig. 2.4 but are slightly flatter over the radius.

Tangential Core and Sidewall Corrections

A unique characteristic of the Beltramian solutions is axial dependence in the tangential velocity. At first, this might thwart the usage of an axially invariant analysis like the one used for the complex-lamellar solution. Fortunately, a closer look at Fig. 2.6b shows the axial dependence is confined only to the bulk inviscid flow region while the solution asymptotes to a functional value of r^{-1} at both the centerline and the sidewall. Therefore, we can ascertain that the boundary layers themselves, in fact, remain axially invariant for these solutions as well. This enables us to start directly with Eq. (2.25) with the coefficient of the convective

term modified to reflect the Beltramian radial velocity. We set

$$\varepsilon\left(\frac{\mathrm{d}^2\xi_\theta}{\mathrm{d}r^2}-\frac{1}{r}\frac{\mathrm{d}\xi_\theta}{\mathrm{d}r}\right)+\kappa\frac{J_1\left(\lambda_0 r\right)}{\beta J_1\left(\lambda_0\beta\right)}\frac{\mathrm{d}\xi_\theta}{\mathrm{d}r}=0 \tag{2.63}$$

where $\xi_\theta = rU_\theta$.

By scaling to the core and expanding the Bessel function as $J_1(\lambda_0\delta s)\approx\lambda_0\delta s/2+\mathcal{O}(\delta^2)$, we find the distinguished limit to be $\delta=\sqrt{\varepsilon/\kappa}$. Substituting these relations into the above equation gives

$$\frac{\mathrm{d}^2\xi_\theta^{(i)}}{\mathrm{d}s^2}+\left[\frac{\lambda_0 s}{2\beta J_1\left(\lambda_0\beta\right)}-\frac{1}{s}\right]\frac{\mathrm{d}\xi_\theta^{(i)}}{\mathrm{d}s}=0 \tag{2.64}$$

with boundary conditions

$$\begin{cases}\xi_\theta^{(i)}(0)=0\\ \lim\limits_{s\to\infty}\xi_\theta^{(i)}=\lim\limits_{r\to 0}\xi_\theta^{(o)}=1\end{cases} \tag{2.65}$$

Fortuitously, we can reclaim a simple closed form solution to this set. After application of the boundary conditions we have

$$\xi_\theta^{(i)}=1-\exp\left[-\frac{\lambda_0 s^2}{4\beta J_1\left(\lambda_0\beta\right)}\right] \tag{2.66}$$

Scaling to the sidewall with $s=(1-r)/\delta$ and expanding the Bessel function like

$$J_1\left(\lambda_0 x\right)\approx\frac{\lambda_0 x}{2}-\frac{\lambda_0^3x^3}{16}\approx-\frac{\lambda_0}{2}\left(\frac{\lambda_0^2}{8}-1\right)+\mathcal{O}\left(\delta\right);\quad x=(1-\delta s) \tag{2.67}$$

finds the sidewall boundary layer equation to be

$$\frac{\mathrm{d}^2\xi_\theta^{(w)}}{\mathrm{d}s^2}+\alpha\frac{\mathrm{d}\xi_\theta^{(w)}}{\mathrm{d}s}=0;\qquad \alpha=\frac{\lambda_0}{2\beta J_1\left(\lambda_0\beta\right)}\left(\frac{\lambda_0^2}{8}-1\right) \tag{2.68}$$

with boundary conditions in the wall domain

$$\begin{cases} \xi_\theta^{(w)}(0) = 0 \\ \lim\limits_{s\to\infty} \xi_\theta^{(w)}(s) = \lim\limits_{r\to 1} \xi_\theta^{(o)}(r) = 1 \end{cases} \tag{2.69}$$

This set leads to the solution

$$\xi_\theta^{(w)} = 1 - \exp\left[-\frac{\lambda_0}{2\beta J_1(\lambda_0\beta)}\left(\frac{\lambda_0^2}{8} - 1\right)s\right] \tag{2.70}$$

Finally, by invoking Eq. (2.37), we can construct the composite solution

$$U_\theta^{(c)} = \frac{1}{r}\left\{\left[1 + \frac{\lambda_0^2\kappa^2 r^2 z^2 J_1^2(\lambda_0 r)}{\beta^2 J_1^2(\lambda_0\beta)}\right]^{1/2} - \exp\left[-\frac{\lambda_0 V r^2}{8\pi\beta J_1(\lambda_0\beta)}\right] \right.$$
$$\left. - \exp\left[-\frac{\lambda_0 V}{4\pi\beta J_1(\lambda_0\beta)}\left(\frac{\lambda_0^2}{8} - 1\right)(1-r)\right]\right\} \tag{2.71}$$

Axial and Radial Corrections

To characterize the axial sidewall boundary layer, we can begin directly with the equation

$$U_r\frac{\partial U_z}{\partial r} + U_z\frac{\partial U_z}{\partial z} = -\frac{\partial p}{\partial z} + \varepsilon\left(\frac{\partial^2 U_z}{\partial r^2} + \frac{1}{r}\frac{\partial U_z}{\partial r}\right) \tag{2.72}$$

Again, we must adhere to the assumptions defined for the tangential velocity. With the advent of the complex-lamellar study, we were inclined to reduce this equation at the present and define a dependent variable transformation later. A more systematic and asymptotically correct procedure is to substitute $u_z = \xi_z(r) z J_0(\lambda_0 r)$ into the governing equation and judiciously truncate the new equation. This step leads to a much more elaborate equation,

namely,

$$\varepsilon\left\{zJ_0\left(\lambda_0 r\right)\xi_z'' - \frac{z\lambda_0^2}{2}\left[J_0\left(\lambda_0 r\right) - J_2\left(\lambda_0 r\right)\right]\xi_z \right.$$
$$\left. -2z\lambda_0 J_1\left(\lambda_0 r\right)\xi_z' + \frac{1}{r}\left[-z\lambda_0 J_1\left(\lambda_0 r\right)\xi_z + zJ_0\left(\lambda_0 r\right)\xi_z'\right]\right\}$$
$$+\kappa\frac{J_1\left(\lambda_0 r\right)}{\beta J_1\left(\lambda_0\beta\right)}\left[-z\lambda_0 J_1\left(\lambda_0 r\right)\xi_z + zJ_0\left(\lambda_0 r\right)\xi_z'\right] = 0$$

In hindsight, this step was also performed with the tangential equation but the result was somewhat obscured by the obvious choice of a variable transformation, $\xi_\theta = rU_\theta$. Upon scaling, we must expand the coefficients above in order to properly identify the leading order terms. We use

$$\begin{cases} J_0\left(\lambda_0 x\right) & \approx 1 - \frac{\lambda_0^2 x^2}{4} \approx -\left(\frac{\lambda_0^2}{4} - 1\right); \quad x = (1-\delta s) \\ J_1\left(\lambda_0 x\right) & \approx \frac{\lambda_0 x}{2} - \frac{\lambda_0^3 x^3}{16} \approx -\frac{\lambda_0}{2}\left(\frac{\lambda_0^2}{8} - 1\right) \\ J_2\left(\lambda_0 x\right) & \approx \frac{\lambda_0^2 x^2}{8} - \frac{\lambda_0^4 x^4}{96} \approx -\frac{\lambda_0^2}{8}\left(\frac{\lambda_0^2}{12} - 1\right) \\ J_3\left(\lambda_0 x\right) & \approx \frac{\lambda_0^3 x^3}{48} - \frac{\lambda_0^5 x^5}{768} \approx -\frac{\lambda_0^3}{48}\left(\frac{\lambda_0^2}{16} - 1\right) \end{cases} \tag{2.73}$$

Clearly, this expansion shows that the Bessel function coefficients have no effect on the ordering in the sidewall region. Backward substitution reveals that the sidewall boundary layer equation is

$$\frac{\mathrm{d}^2\xi_z^{(w)}}{\mathrm{d}s^2} + \alpha\frac{\mathrm{d}\xi_z^{(w)}}{\mathrm{d}s} = 0; \qquad \alpha = \frac{\lambda_0}{2\beta J_1\left(\lambda_0\beta\right)}\left(\frac{\lambda_0^2}{8} - 1\right) \approx 4.91131 \tag{2.74}$$

Applying similar boundary conditions to those used in the tangential direction and building the composite solution completes this derivation. We get

$$U_z^{(c)} = \frac{\lambda_0\kappa z J_0\left(\lambda_0 r\right)}{\beta J_1\left(\lambda_0\beta\right)}\left\{1 - \exp\left[-\frac{\lambda_0 V}{4\pi\beta J_1\left(\lambda_0\beta\right)}\left(\frac{\lambda_0^2}{8} - 1\right)(1-r)\right]\right\} \tag{2.75}$$

The same analysis may be applied in the radial direction to the extent of obtaining

$$\varepsilon\left(\frac{\partial^2 U_r}{\partial r^2}+\frac{1}{r}\frac{\partial U_r}{\partial r}-\frac{U_r}{r^2}\right)+\kappa\frac{J_1\left(\lambda_0 r\right)}{\beta J_1\left(\lambda_0\beta\right)}\frac{\partial U_r}{\partial r}=\mathcal{O}(\kappa^2) \tag{2.76}$$

No deviations from the previous analysis exist here. Given that the complete details may be found in the original paper [28], we jump to the result and write

$$U_r^{(c)}=-\frac{\kappa J_1\left(\lambda_0 r\right)}{\beta J_1\left(\lambda_0\beta\right)}\left\{1-\exp\left[-\frac{\lambda_0 V}{4\pi\beta J_1\left(\lambda_0\beta\right)}\left(\frac{\lambda_0^2}{8}-1\right)(1-r)\right]\right\} \tag{2.77}$$

From the Euler equations, the pressure profile is found to be

$$\begin{aligned}\Delta P\left(r,z\right)=&\frac{r^2-1}{2r^2}+\frac{1}{2}e^{-\frac{V}{\pi}\gamma}-e^{-\frac{V}{2\pi}\gamma}+\frac{1}{r^2}\left(e^{-\frac{V}{2\pi}\gamma r^2}-\frac{1}{2}e^{-\frac{V}{\pi}\gamma r^2}\right)\\&+\frac{V}{2\pi}\gamma\left[\mathrm{Ei}\left(-\frac{V}{\pi}\gamma\right)-\mathrm{Ei}\left(-\frac{V}{\pi}\gamma r^2\right)+\mathrm{Ei}\left(-\frac{V}{2\pi}\gamma r^2\right)-\mathrm{Ei}\left(-\frac{V}{2\pi}\gamma\right)\right]\\&+\frac{1}{2}+\frac{V}{2\pi}\alpha-\frac{V^2}{2\pi^2}\alpha^2\mathrm{Ei}\left(\frac{V}{\pi}\alpha\right)e^{-\frac{V}{\pi}\alpha}\\&+\frac{1}{2r^2\varepsilon^2}\left(-e^{-\frac{V}{\pi}\alpha(1-r)}\varepsilon\left(\varepsilon+2r\alpha\kappa\right)+4r^2\alpha^2\kappa^2e^{-\frac{V}{\pi}\alpha}\mathrm{Ei}\left(\frac{V}{\pi}\alpha r\right)\right)\\&-\frac{4\pi^2\kappa^2+2\pi\alpha V\kappa^2-\alpha^2\kappa^2\mathrm{Ei}\left(\frac{V}{2\pi}\alpha\right)e^{-\frac{V}{2\pi}\alpha}}{4\pi^2\kappa^2}\\&+\frac{\left(4\pi^2\kappa^2+2\pi V\kappa^2\alpha r\right)e^{-\frac{V}{2\pi}\alpha(1-r)}-r^2V^2\alpha^2\kappa^2e^{-\frac{V}{2\pi}\alpha}\mathrm{Ei}\left(\frac{V}{2\pi}\alpha r\right)}{4\pi^2\kappa^2r^2}\\&+\frac{z^2\kappa^2\lambda_0^2}{2\beta^2J_1\left(\lambda_0\beta\right)}\left[J_0^2\left(\lambda_0\right)-J_0^2\left(\lambda_0 r\right)+J_1^2\left(\lambda_0\right)-J_1^2\left(\lambda_0 r\right)\right]\end{aligned} \tag{2.78}$$

Figure 2.8 illustrates the uniformly valid viscous profile for the linear Beltramian bidirectional vortex.

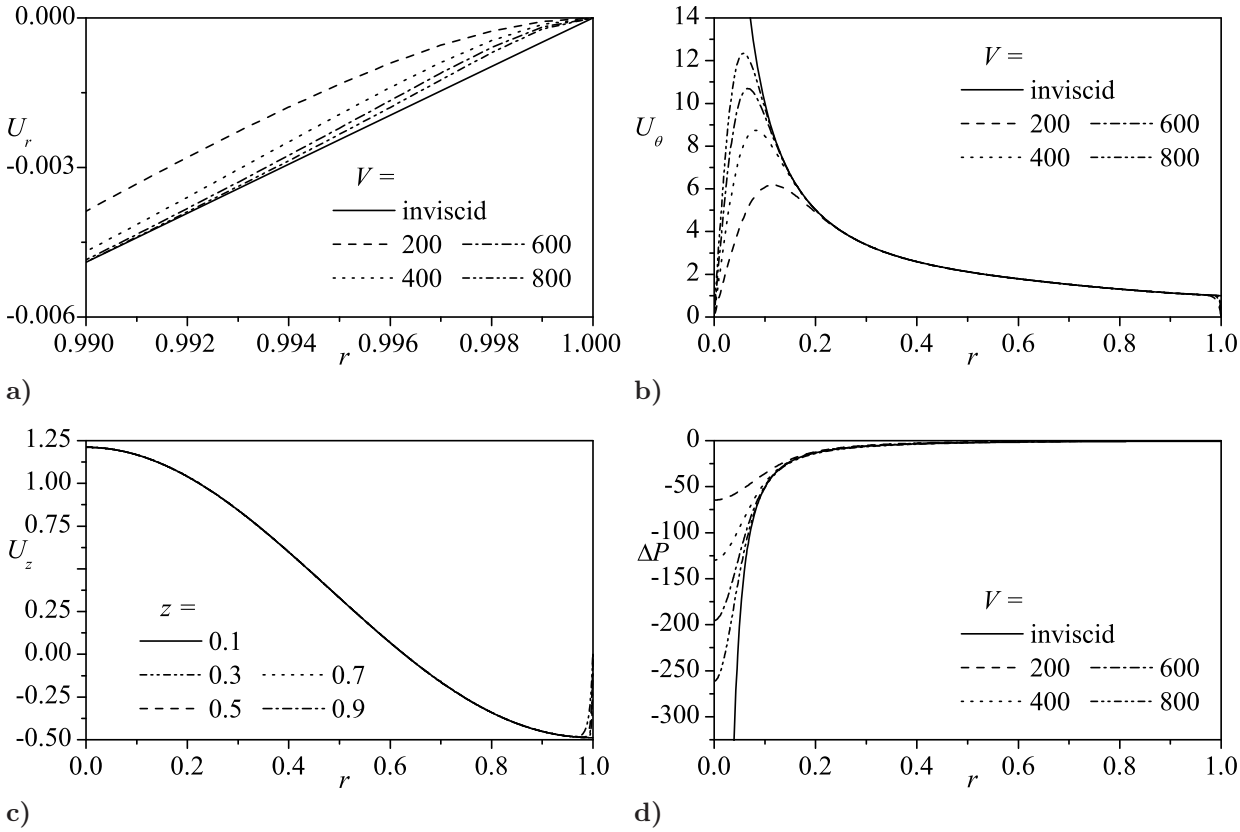


Figure 2.8: The viscous linear Beltramian bidirectional vortex with $\kappa = 0.103$ and $z = 1$.

2.3.3 The Harmonic Beltramian Solution

The sinusoidal axial dependence is generated by taking $+\upsilon^2$ in Eq. (2.57). For this case, the general solution is found to be [26]

$$\psi = r\left[C_1 \sin(\upsilon z) + C_2 \cos(\upsilon z)\right]\left[C_3 J_1\left(r\sqrt{C_m^2 - \upsilon^2}\right) + C_4 Y_1\left(r\sqrt{C_m^2 - \upsilon^2}\right)\right]; \quad C_m^2 > \upsilon^2 \tag{2.79}$$

We refer to the boundary conditions given by Eq. (2.16) with the addition of $U_r(r, l) = 0$ to procure a value for υ. The first two boundary conditions suggest $C_2 = C_4 = 0$ and result in

$$\psi = \psi_0 \sin(\upsilon z) r J_1\left(r\sqrt{C_m^2 - \upsilon^2}\right); \quad \psi_0 = C_1 C_3 \tag{2.80}$$

The third constraint identifies the eigenvalue equation, $J_1(\sqrt{C_m^2 - \upsilon^2}) = 0$, wherefrom $C_m^2 = {\lambda_m}^2 + \upsilon^2$ may be deduced and λ_m denotes the mth root of the Bessel function of the first kind. These are given by the standard sequence, $\lambda_m = \{3.83171, 7.01559, 10.1735, 13.3237, ...\}$. For bidirectional motion, we only Here, $\lambda_m = \lambda_0$. The streamfunction collapses into

$$\psi = \psi_0 \sin(\upsilon z)\, r J_1(\lambda_0 r)\,; \quad C_0^2 > \upsilon^2 \tag{2.81}$$

Our additional boundary condition can now be employed to determine the separation constant, υ. We find $\cos(\upsilon_j l) = 0$ where $\upsilon_j = (j + \frac{1}{2})\pi/l$. To preclude recirculation within the interval $0 \leq z \leq l$, we consider only the case of

$$\upsilon_j = \upsilon_0 = \frac{\pi}{2l}; \quad C_0^2 = \lambda_0^2 + \frac{\pi^2}{4l^2} \tag{2.82}$$

Other flow patterns associated with $j = 1, 2, 3...$ or $z > l$ will generate axially periodic vortex cells that do not correspond to the problem under investigation.

Lastly, the boundary condition requiring conservation of volumetric flow rate is used to determine that $\psi_0 = Q_i/[2\pi\beta J_1(\lambda_0\beta)]$. At length, we collect

$$\psi = \kappa l r \sin\left(\tfrac{1}{2}\pi z/l\right) \frac{J_1(\lambda_0 r)}{\beta J_1(\lambda_0\beta)}; \quad C_0^2 > \upsilon^2 \tag{2.83}$$

where $\kappa = Q_i/(2\pi l)$.

Finally, we are able to express the complete inviscid profile as

$$\begin{aligned} \boldsymbol{U} = &-\frac{\pi}{2}\kappa r \cos\left(\tfrac{1}{2}\pi z/l\right) \frac{J_1(\lambda_0 r)}{\beta J_1(\lambda_0\beta)}\, \boldsymbol{e}_r \\ &+ r^{-1}\sqrt{1 + \frac{\kappa^2}{\beta^2 J_1^2(\lambda_0\beta)}\left(\lambda_0^2 l^2 + \frac{1}{4}\pi^2\right) r^2 \sin^2\left(\tfrac{1}{2}\pi z/l\right) J_1^2(\lambda_0 r)}\, \boldsymbol{e}_\theta \\ &+ \kappa l \lambda_0 \sin\left(\tfrac{1}{2}\pi z/l\right) \frac{J_0(\lambda_0 r)}{\beta J_1(\lambda_0\beta)}\, \boldsymbol{e}_z \end{aligned} \tag{2.84}$$

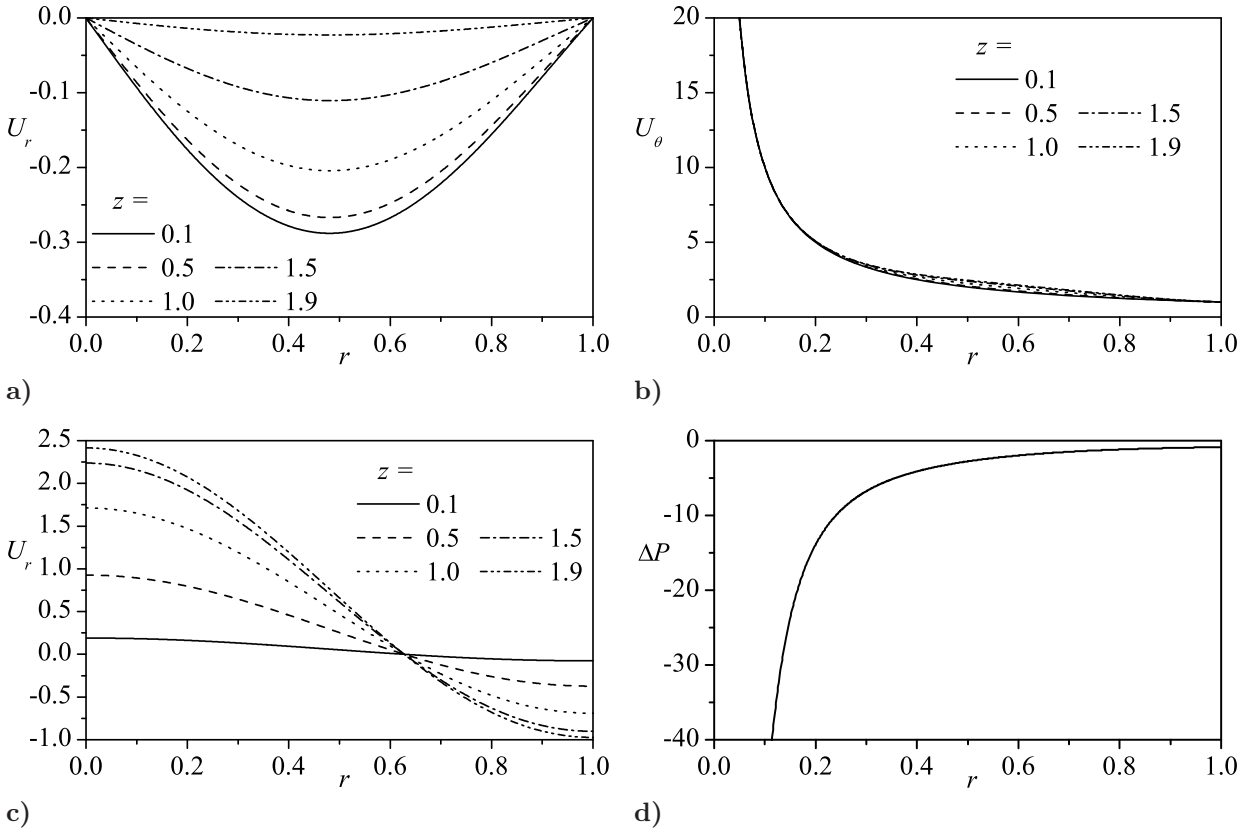


Figure 2.9: The inviscid harmonic Beltramian bidirectional vortex with $\kappa = 0.103$, $l = 2$, and $z = 0.3$.

Figures 2.9–2.10 show similar profiles to the linear Beltramian model with the exception of the new, axially dependent, radial velocity. The complete discussion by Batterson and Majdalani [29] explore this similarity by extending the study further through evaluation over a variety of flow characteristics.

To include a complete viscous treatment of the harmonic Beltramian solution would be a lesson in redundancy. We should, however, discuss how we handle the axial dependence in the radial solution and its role as a viable coefficient in the convective boundary layer term. The inclusion of U_r as a convective coefficient reintroduces axial dependence into the governing equation. This is contrary to our original assumptions and must be handled carefully. It could be speculated that the core vortex in this case may be sensitive to variations in the

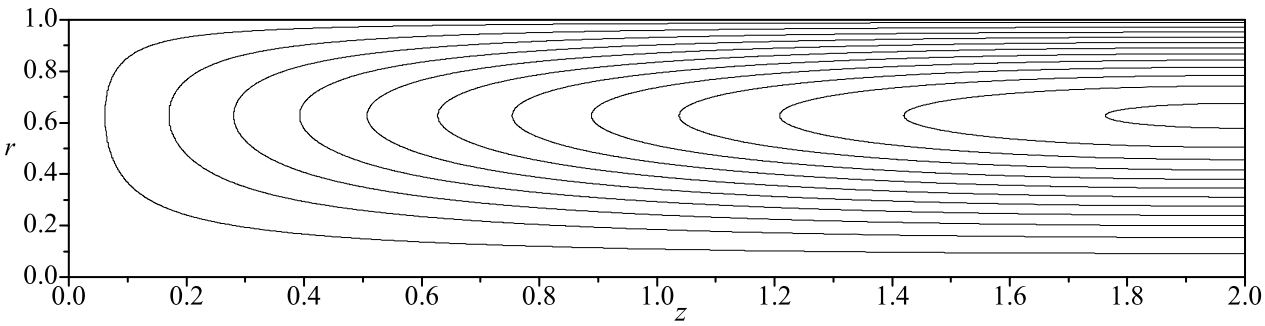


Figure 2.10: Streamlines of the harmonic Beltramian BV.

crossflow velocity. Nonetheless, we also recall that for an axisymmetric flowfield, continuity provides the link between the axial and radial velocities only; the tangential velocity is not affected by crossflow velocity through continuity. Hence, we can Here, U_θ to be unaffected by an axially varying u_r. In order to make headway, we adopt an average value of the radial velocity in the core region by taking

$$(U_r)_{\text{avg}} = -\frac{\pi\kappa J_1(\lambda_0 r)}{2\beta J_1(\lambda_0\beta)} l^{-1} \int_0^l \cos\left(\tfrac{1}{2}\pi z/l\right) = -\frac{\kappa J_1(\lambda_0 r)}{\beta J_1(\lambda_0\beta)} \tag{2.85}$$

Henceforth, the governing equations for all vector directions are reduced to those discovered in the linear Beltramian analysis and given in Eq. (2.63). The boundary layer character between the two Beltramian solutions is the same. Its matching to the outer, inviscid, solution accounts for small differences in the composite solution. At length, the

harmonic Beltramian solution is shown to possess the viscous character [29]

$$\begin{aligned}\boldsymbol{U} = -\frac{\pi\kappa}{2\beta J_1(\lambda_0\beta)}\cos\left(\tfrac{1}{2}\pi z/l\right) J_1(\lambda_0 r)\times \\ \left\{1-\exp\left[-\frac{V}{2\pi}\alpha(1-r)\right]\right\}\boldsymbol{e}_r \\ r^{-1}\left\{\left[1+\frac{\kappa^2}{\beta^2 J_1^2(\lambda_0\beta)}\left(\lambda_0^2 l^2+\frac{1}{4}\pi^2\right) r^2\sin^2\left(\tfrac{1}{2}\pi z/l\right) J_1^2(\lambda_0 r)\right]^{1/2}\right. \\ \left.-\exp\left(-\frac{V}{2\pi}\gamma r^2\right)-\exp\left[-\frac{V}{2\pi}\alpha(1-r)\right]\right\}\boldsymbol{e}_\theta \\ +\frac{\lambda_0\kappa l}{\beta J_1(\lambda_0\beta)}\sin\left(\tfrac{1}{2}\pi z/l\right) J_0(\lambda_0 r)\times \\ \left\{1-\exp\left[-\frac{V}{2\pi}\alpha(1-r)\right]\right\}\boldsymbol{e}_z \end{aligned} \quad (2.86)$$

with the associated pressure profile

$$\begin{aligned}\Delta P(r,z) = \frac{r^2-1}{2r^2}-\frac{V}{2\pi}\alpha+\frac{\pi^2+\pi\alpha V-\alpha^2V^2\mathrm{Ei}\left(\frac{V}{\pi}\alpha\right)e^{-\frac{V}{\pi}\alpha}}{2\pi^2} \\ -\frac{\pi(\pi+V\alpha r)e^{-\frac{V}{\pi}\alpha(1-r)}-r^2\alpha^2V^2\mathrm{Ei}\left(\frac{V}{\pi}\alpha r\right)e^{-\frac{V}{\pi}\alpha}}{2\pi^2r^2} \\ +\frac{1}{r^2}\left(e^{-\frac{V}{2\pi}\gamma r^2}-\frac{1}{2}e^{-\frac{V}{\pi}\gamma r^2}\right)+\frac{\pi}{2\pi}\left(e^{-\frac{V}{\pi}\gamma}-2e^{-\frac{V}{2\pi}\gamma}\right)+\frac{V\gamma}{2\pi}\left[\mathrm{Ei}\left(-\frac{V}{\pi}\gamma\right)-\mathrm{Ei}\left(-\frac{V}{\pi}\gamma r^2\right)\right] \\ +\frac{V\gamma}{2\pi}\left[\mathrm{Ei}\left(-\frac{V}{2\pi}\gamma r^2\right)-\mathrm{Ei}\left(-\frac{V}{2\pi}\gamma\right)\right]+\frac{\alpha^2V^2}{4\pi^2}\mathrm{Ei}\left(\frac{V}{2\pi}\alpha\right)e^{-\frac{V}{2\pi}\alpha} \\ +\frac{2\pi(2\pi+V\alpha r)e^{-\frac{V}{2\pi}\alpha(1-r)}-r^2\alpha^2V^2\mathrm{Ei}\left(\frac{V}{2\pi}\alpha r\right)e^{-\frac{V}{2\pi}\alpha}}{4r^2\pi^2} \end{aligned} \quad (2.87)$$

Figure 2.11 shows the complete viscous solution. Take note to the glaring similarity to its linear counterpart in Fig. 2.8.

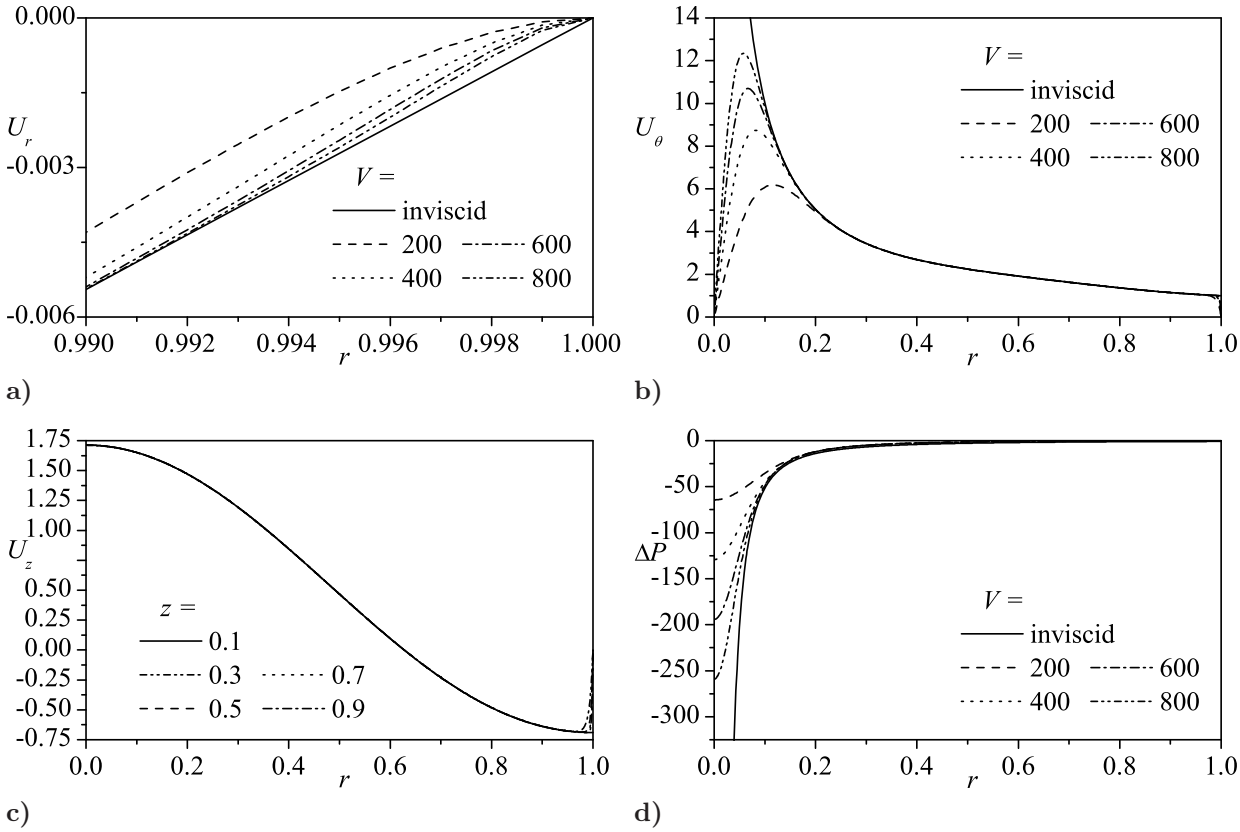


Figure 2.11: The viscous harmonic Beltramian bidirectional vortex with $\kappa = 0.103$, $l = 2$, and $z = 1$.

2.4 A Comment on the Existence of Multiple Mantles

Experimental and numerical evidence gathered by Anderson *et al.* [31], Rom, Anderson and Chiaverini [32], and others have corroborated the existence of interchanging flow reversals. The presence of multi-layering is evident and has been aforementioned by identifying higher eigensolutions in previous sections and from Vyas and Majdalani's complex-lamellar model [30].

Bidirectional flow is spurred on by the choice of $\lambda_m = \lambda_0 = 3.832$. Each consecutive eigenvalue will cause one additional flow reversal. For propulsive applications we require the flow to exit at the aft end of the chamber. Only even eigenvalues with $m = 0, 2, 4, \ldots$

Table 2.1: Eigenvalues and corresponding mantle locations for even flow reversal mode numbers and an odd number of internal mantles.

m	λ_m	$\beta_{m,0}$	$\beta_{m,1}$	$\beta_{m,2}$	$\beta_{m,3}$	$\beta_{m,4}$	$\beta_{m,5}$	$\beta_{m,6}$
0	π	0.707						
2	3π	0.408	0.707	0.913				
4	5π	0.316	0.548	0.707	0.837	0.949		
6	7π	0.267	0.463	0.598	0.707	0.802	0.886	0.964

a) Complex-lamellar mantles

m	λ_m	$\beta_{m,0}$	$\beta_{m,1}$	$\beta_{m,2}$	$\beta_{m,3}$	$\beta_{m,4}$	$\beta_{m,5}$	$\beta_{m,6}$
0	3.832	0.628						
2	10.174	0.236	0.543	0.851				
4	16.471	0.146	0.335	0.525	0.716	0.907		
6	22.760	0.106	0.243	0.380	0.518	0.656	0.794	0.932

a) Beltramian mantles

may be considered lest an unphysical setting with implausible inflow and outflow boundary conditions is returned. Further evidence will be shown to corroborate this ascertation.

For an ideal configuration in which no collisions occur during outflow, the exit port radius may be chosen in such a way to match the position of the innermost mantle, for any given flow reversal mode number, m. The flow configuration associated with each increasing eigenvalue may be linked to a progressively shorter mantle radius, $\beta_{m,0}$ $\{m = 0, 2, 4, \cdots\}$. For a fixed reversal mode number m, the locations of all internal mantles $\beta_{m,n}$ $\{n = 0, 1, \cdots, m\}$ may be extracted from the roots of $J_1(\lambda_m \beta_{m,n}) = 0$ and catalogued in Table 2.1. We must note that, unlike the complex-lamellar model in which the outlet radius does not appear as a parameter (where $\beta = 1/\sqrt{2}$), it is important to retain $\beta_{m,0}$ in the Beltramian solutions. However, in Table 2.1 we denote the respective mantle locations in the same way for both models.

Although the complete derivation is not shown, viscous corrections to the complex-lamellar multidirectional vortex first appears in the paper by Batterson and Majdalani [33].

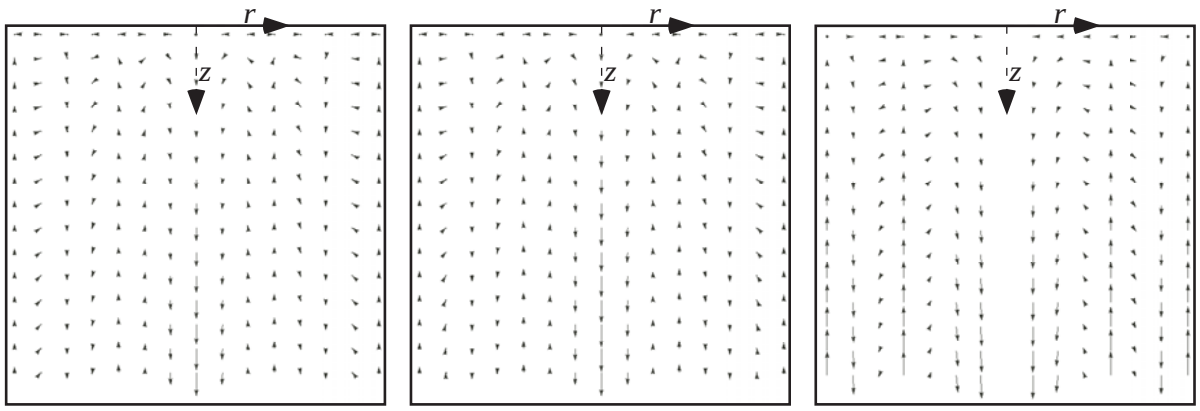
Following a similar analysis they arrive at

$$\boldsymbol{u} = -\frac{\kappa}{r}\sin[(m+1)\pi r^2]\left[1 - e^{-\frac{V}{4}(m+1)\alpha(1-r^2)}\right]\boldsymbol{e}_r + r^{-1}\left[1 - e^{-\frac{V}{4}(m+1)r^2} - e^{-\frac{V}{4}(m+1)\alpha(1-r^2)}\right]\boldsymbol{e}_\theta + 2(m+1)\pi\kappa z\cos[(m+1)\pi r^2]\left[1 - e^{-\frac{V}{4}(m+1)\alpha(1-r^2)}\right]\boldsymbol{e}_z$$

where $\alpha = \frac{1}{6}(m+1)^2\pi^2 - 1$ may be associated with the complex-lamellar solution. A careful comparison of the three multidirectional vortex models resides in the recent paper by Batterson and Majdalani [33].

As for the Beltramian models, no significant modification needs to be made in the mathematical formulation itself to characterize multi-directional flow. One may simply put, $\lambda_0 \to \lambda_m$ and $\beta \to \beta_{m,0}$. This applies equally well to the definitions of γ and α. We have for the linear case:

$$\boldsymbol{U} = -\frac{\kappa J_1(\lambda_m r)}{\beta J_1(\lambda_m \beta_{m,0})}\left[1 - e^{-\frac{V}{2\pi}\alpha(1-r)}\right]\boldsymbol{e}_r + \frac{1}{r}\left[\left(1 + \frac{\lambda_m^2\kappa^2 r^2 z^2 J_1^2(\lambda_m r)}{\beta_{m,0}^2 J_1^2(\lambda_m\beta_{m,0})}\right)^{1/2} - e^{-\frac{V}{2\pi}\gamma r^2} - e^{-\frac{V}{2\pi}\alpha(1-r)}\right]\boldsymbol{e}_\theta + \frac{\lambda_m\kappa z J_0(\lambda_m r)}{\beta_{m,0}J_1(\lambda_m\beta_{m,0})}\left[1 - e^{-\frac{V}{2\pi}\alpha(1-r)}\right]\boldsymbol{e}_z \quad (2.88)$$



a) Linear Beltramian **b)** Harmonic Beltramian **c)** Complex Lamellar

Figure 2.12: Vector plots of a) linear Beltramian, b) harmonic Beltramian, and c) complex-lamellar models. Here $l = 2$.

and for the harmonic case:

$$\begin{aligned}\boldsymbol{U} = &-\frac{\pi\kappa}{2\beta_{m,0}J_1\left(\lambda_m\beta_{m,0}\right)}\cos\left(\frac{\pi z}{2l}\right)J_1\left(\lambda_m r\right)\left(1-e^{-\frac{V}{2\pi}\alpha(1-r)}\right)\boldsymbol{e}_r \\ &+\frac{1}{r}\left\{\left[1+\frac{\kappa^2}{\beta_{m,0}^2J_1^2\left(\lambda_m\beta_{m,0}\right)}\left(\lambda_m^2l^2+\tfrac{1}{4}\pi^2\right)r^2\sin^2\left(\frac{\pi z}{2l}\right)J_1^2\left(\lambda_m r\right)\right]^{1/2}\right. \\ &\qquad\left.-e^{-\frac{V}{2\pi}\gamma r^2}-e^{-\frac{V}{2\pi}\alpha(1-r)}\right\}\boldsymbol{e}_\theta \\ &\qquad+\frac{\lambda_m\kappa l}{\beta_{m,0}J_1\left(\lambda_m\beta_{m,0}\right)}\sin\left(\frac{\pi z}{2l}\right)J_0\left(\lambda_m r\right)\left(1-e^{-\frac{V}{2\pi}\alpha(1-r)}\right)\boldsymbol{e}_z \qquad (2.89)\end{aligned}$$

where $\alpha = \lambda_m\left(\frac{1}{8}\lambda_m^2-1\right)/[2\beta_{m,0}J_1(\lambda_m\beta_{m,0})]$ and $\gamma = \lambda_m/[4\beta_{m,0}J_1(\lambda_m\beta_{m,0})]$.

In Fig. 2.12, vector diagrams of the three models are displayed. These graphs show that a higher centerline velocity persists farther downstream in the nonlinear profile than in any other model. The vector directions also suggest a headwall-skewed crossflow along the mantle surface in the nonlinear solution. This leads to higher crossflow velocities near the headwall, and reduced spillage near the endwall. Such behavior is quite advantageous as it is more desirable to limit mass transport near $z = l$ from the outer vortex into the inner core. It also serves to satisfy the purely tangential injection boundary condition.

2.4.1 Experimental Validation

The available empirical measurements suggest that the Beltramian model outperforms other models, including numerical simulations, in predicting the location of the internal mantles. The empirical results in Table 2.2 are taken from Anderson, Rom and coworkers [31, 32, 93]. These seem to suggest that in the presence of three mantles. We see that the original complex-lamellar model is less precise than the Beltramian solution in capturing mantle locations. Without knowledge of the Beltramian formulation, Rom, Anderson and Chiaverini note that the complex-lamellar model *with four mantles* matches quite remarkably their experimental and computational predictions. However, based on the present investigation, it does not appear that four mantles are likely to form without moving the primary injection plane to the headwall. Instead, it is our belief that the four-mantle case reported by Anderson and coworkers actually corresponds to a three-mantle configuration with $m = 2$. The confusion could be attributed to poor resolution of the experimental measurements in the vicinity of the sidewall. In Table 2.2, it can be seen that the Beltramian model concurs reasonably well with both experimental and computation results. It is therefore recommended for use in modeling multidirectional vortex behavior. Unfortunately, there cannot be any further differentiation between the physicality of the linear and harmonic models at the time of this writing.

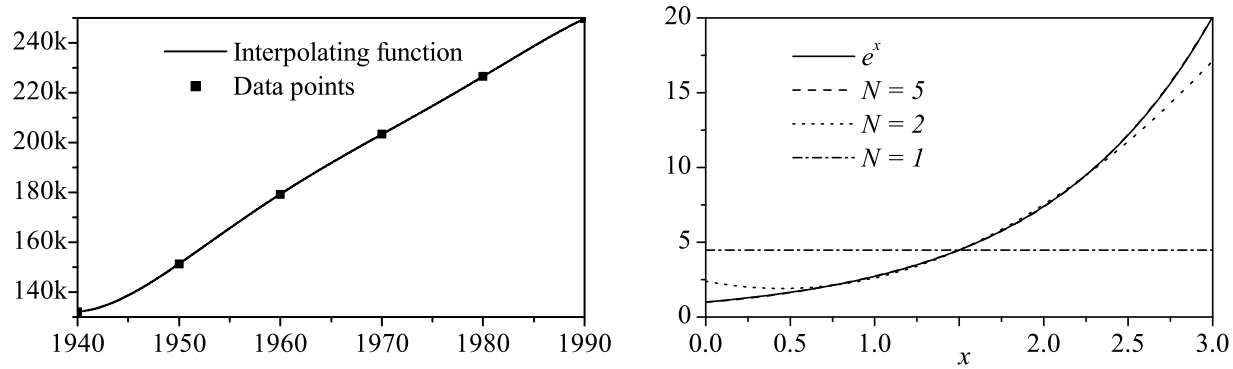
Table 2.2: Comparison of experimental and computational mantle locations with the complex-lamellar (CL) and Beltramian (BEL) models [31, 32, 93].

			Experimental			Computational		
n	$\beta_{2,n}^{(\mathrm{CL})}$	$\beta_{2,n}^{(\mathrm{BEL})}$	$\beta_{2,n}^{(\mathrm{EXP})}$	$\left\lvert\beta_{2,n}^{(\mathrm{CL})} - \beta_{2,n}^{(\mathrm{EXP})}\right\rvert$	$\left\lvert\beta_{2,n}^{(\mathrm{BEL})} - \beta_{2,n}^{(\mathrm{EXP})}\right\rvert$	$\beta_{2,n}^{(\mathrm{CFD})}$	$\left\lvert\beta_{2,n}^{(\mathrm{CL})} - \beta_{2,n}^{(\mathrm{CFD})}\right\rvert$	$\left\lvert\beta_{2,n}^{(\mathrm{BEL})} - \beta_{2,n}^{(\mathrm{CFD})}\right\rvert$
0	0.408	0.236	0.296	0.112	0.060	0.305	0.103	0.069
1	0.707	0.543	0.594	0.113	0.051	0.385	0.322	0.158
2	0.913	0.851	0.803	0.110	0.048	0.787	0.126	0.064

Chapter 3

Spectral Collocation Methods

In this section we will introduce the mathematical aspects required to construct the hydrodynamic instability problem. To do so, we will formulate the fundamental mathematical basis for spectral methods applied to one and two-dimensional differential equations. Likewise we will show the extension of these concepts to numerical programming in MATLAB. To some extent this section could be reduced or even eliminated. However, its inclusion is threefold. First, it gives validity to the other sections by showing the exact, systematic methodology used to handle instability problems. Second, collocation methods are extremely powerful yet only a few publications present their application in a clear, concise manner. A review of collocation schemes is therefore in order, especially that the development of a complete spectral stability code is far from trivial. As we shall see, although the fundamental mathematics behind spectral solvers is not new, their adaptation into functional codes in the context of instability analysis is still under development. Lastly, the usefulness of spectral methods in numerical modeling cannot be overstated and this section puts their full range of uses on display.



a) Interpolation of census data (in thousands) [94].

b) Lagrange interpolating polynomial approximations of the function e^x for different resolutions.

Figure 3.1: Two examples of interpolating polynomials.

3.1 Polynomial Approximation and Interpolation

Consider the example of a population census taken every ten years. The exact population is known only at ten year intervals while the population on off years must be speculated. It would be helpful to use a function that smoothly fits the given data in order to have a continuous census record without tabulating populations every year. This process is called *interpolation* and is often done through the construction of *interpolating polynomials*. Figure 3.1 shows two examples of interpolation. Figure 3.1a follows up on the problem of population growth while Fig. 3.1b shows the Lagrange interpolating polynomial approximation for the function e^x for several degrees of spectral resolution. Clearly, the latter is more relevant to the hydrodynamic stability problem in that interpolating polynomials are introduced to approximate the unknown functional solution to the governing equations.

The choice of the type of polynomial used is often problem specific and driven by the need for localized grid resolution, numerical stability, and accuracy. Algebraic polynomials (Power or Maclaurin series) can be constructed in a variety of ways including Lagrange Polynomials, Newton Divided Differences, Padé Approximations, or even straightforward Taylor Series Expansions of known functions [94]. Trigonometric polynomial interpolators

and other orthogonal polynomials such as the set of Chebyshev or Legendre polynomials can be particularly advantageous for analytic problems involving quadrature and differential equations. Furthermore, there are distinct advantages of using nonuniformly spaced interpolating (collocation) points such as Chebyshev (or Gauss-Lobatto) points rather than equally spaced points. Periodic and equally spaced points tend toward Gibb's phenomenon for Fourier spectral methods and Runge phenomenon for algebraic methods such as Lagrange interpolation. Likewise, if we were to differentiate an equispaced polynomial to find a derivative approximation, then the errors should be expected to propagate accordingly. Figure 3.2 and Alg. B.1.1 on page 294 reproduce the example by Trefethen [95]. It clearly shows the deficiencies found with simple algebraic (equally spaced) polynomial interpolation. It also identifies the advantages of Chebyshev spectral methods when requiring high resolution near the boundaries. Boundary layer theory in fluid mechanics is a prime candidate for Chebyshev grids.

Unlike interpolation for experimental data points, interpolating polynomials constructed to solve equations are devised to match the exact solution at specific locations called *collocation points*. Collocation points coincide with the zeros of the interpolating polynomial. In the case of Chebyshev polynomials, the collocation points are actually the Gauss-Lobotto nodes.

3.2 Chebyshev Polynomials

Several trigonometric or orthogonal functions are used in interpolation algorithms. Chebyshev polynomials of the first type satisfy the equation

$$(1-\xi^2)T''_{N-1}(\xi) - \xi T'_{N-1}(\xi) + N^2 T_{N-1}(\xi) = 0 \tag{3.1}$$

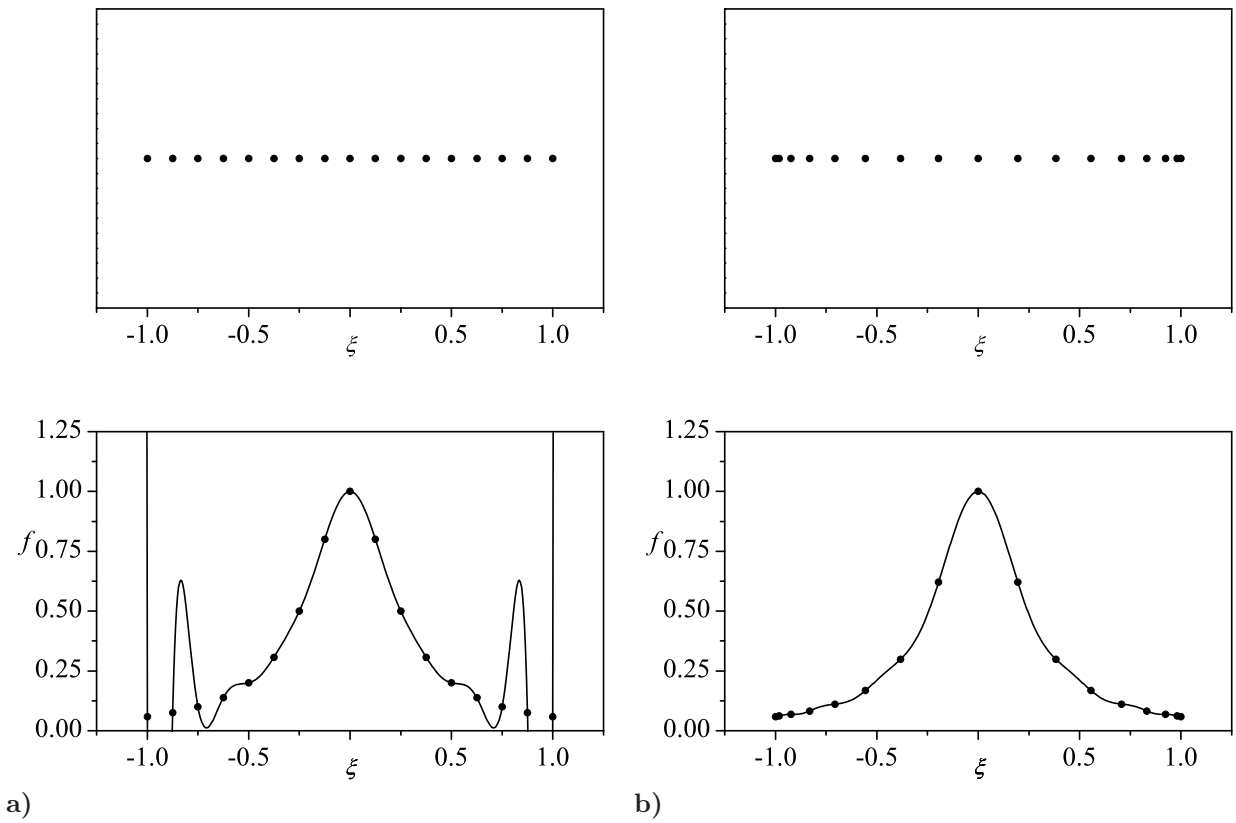


Figure 3.2: Comparison between interpolation with equally spaced (left) and Chebyshev grids (right). The Runge phenomenon can be easily seen.

Its solution is facilitated by appling a change of variables, namely,

$$\xi = \cos\theta \quad \text{with} \quad \frac{\mathrm{d}}{\mathrm{d}\xi} = \frac{-1}{\sin\theta}\frac{\mathrm{d}}{\mathrm{d}\theta} \tag{3.2}$$

resulting in

$$\frac{\mathrm{d}^2 T_{N-1}}{\mathrm{d}\theta^2} + (N-1)^2 T_{N-1} = 0 \tag{3.3}$$

This is a simple harmonic oscillator with the solution $T_{N-1}(\theta) = \cos[(N-1)\theta] = \cos\left[(N-1)\arccos\xi\right]$. We see from this form that Chebyshev polynomials should therefore be orthogonal. This is actually only true in the interval $[-1, 1]$ (see [96]).

As we will see, it is also necessary to know the general derivative formula for these polynomials. Consider

$$T_{N-1}(\xi) = T_{N-1}\left[\cos(\theta)\right] = \cos[(N-1)\theta]; \quad \forall\theta \tag{3.4}$$

Differentiation with respect to θ is then

$$\frac{\mathrm{d}T_{N-1}}{\mathrm{d}\theta} = -T'_{N-1}(\cos\theta)\sin\theta = -(N-1)\sin[(N-1)\theta] \tag{3.5}$$

Finally, we solve for T'_{N-1}, where the prime denotes differentiation with respect to ξ:

$$T'_{N-1}(\theta) = (N-1)\frac{\sin[(N-1)\theta]}{\sin(\theta)} \quad \text{for} \quad N = 1, 2, 3... \quad \text{with} \quad \theta = \arccos\xi \tag{3.6}$$

This result is important for spectral representation of derivatives using Chebyshev polynomials.

As a side note, this formula connects Chebyshev polynomials of the first and second kind in that

$$T'_{N-1}(\theta) = (N-1)U_{N-1}(\theta) \tag{3.7}$$

with $U_N(\theta)$ being defined as the Chebyshev polynomial of the second kind.

From a more practical perspective, polynomials can be used to approximate known and unknown functions to degree N. If we define the polynomial representation of degree infinity of a function $f(\xi)$ as $\mathcal{P}_\infty f(\xi_\infty)$ then we can step back and define a discrete polynomial representative of a function, $f(\xi)$, at order N as

$$\mathcal{P}_N f(\xi_N) = \sum_{i=1}^{N} f(\xi_i)\lambda_i(\xi) \tag{3.8}$$

Table 3.1: Example collocation points, ξ_i for increasing polynomial order, N.

N	$i =$	1	2	3	4	5
1	$\xi_i =$	0				
2		1	-1			
3		1	0	-1		
4		1	0.5	-0.5	-1	
5		1	0.7071	0	-0.7071	-1

where ξ_i represents the collocation points from $i = 1, \ldots, N$ and $\lambda_i(\xi)$ defines the weight functions. The collocation points for the first six Chebyshev polynomials are shown in Table 3.1.

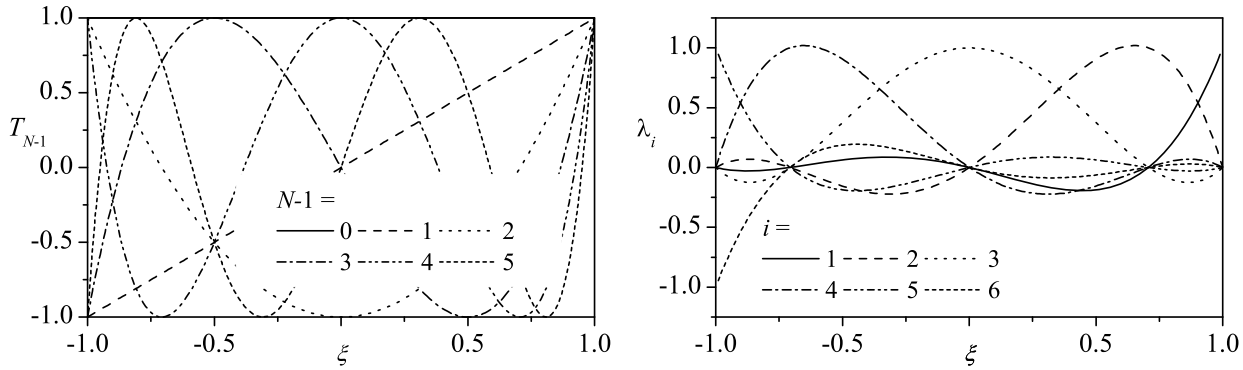
For Chebyshev polynomial interpolation, the weight function takes the form

$$\lambda_i(\xi) = (-1)^i \left(\frac{1-\xi^2}{\xi - \xi_i}\right) \left[\frac{T'_{N-1}(\theta)}{d_i(N-1)^2}\right] \quad \text{with} \quad \begin{cases} \xi_i = \cos[\frac{(i-1)\pi}{N-1}] \\ \theta = \arccos \xi \end{cases} \tag{3.9}$$

with

$$d_i = \begin{cases} 2; & i = 1 \text{ or } N \text{ (endpoints)}, \\ 1; & \text{otherwise (interior points)} \end{cases} \tag{3.10}$$

Equation (3.9) leads to a system of N equations and N unknowns to be solved simultaneously in order to find the discrete values of the function f at the collocation points ξ_i. The distribution of xi_i is depicted in Fig. 3.2 for $N = 17$. The resulting polynomial solution is constructed by multiplying the weight functions by the functional value at the collocation points. Analytically, this is unreasonable for high resolution solutions but is well suited for numerical computation. Although time consuming for large systems, Gaussian elimination is the most available technique. Fortunately the algorithm `chebint` provided by Weideman and Reddy [97] will interpolate a coarse solution over a finer grid. It reliably interpolates over the collocation points and develops the final polynomial solution according to the *Barycentric*



a) Chebyshev polynomials, $T_N - 1$

b) Chebyshev interpolating polynomial weight functions, λ_i

Figure 3.3: Chebyshev polynomials and corresponding weight functions for $N = 5$.

Form of the interpolant [98]. This function is provided in App. B.4.1. Figure 3.3 depicts the first six Chebyshev polynomials and the discrete weight function for sixth order polynomial interpolation.

3.3 Pseudo-Spectral Derivatives

When considering collocation methods for differential equations, we must be able to represent the derivatives of functions accurately and efficiently. Classic finite difference methods or divided differences are often used to compute derivative approximations. This application is practical and straightforward, but leads one to potentially unstable calculations and sensitivity to collocation step size. For example, if Lagrange interpolation is used, equally spaced collocation nodes return unstable approximations for the first derivative of function f for large N [99]. Therefore, the natural alternative is to use a nonuniformly distributed scheme akin to a Chebyshev collocation.

If we recall our polynomial approximation defined as $\mathcal{P}_N f$, then we can approximate the first derivative of function f at the collocation nodes, $\xi_1, \ldots, \xi_N$, with the exact first derivative of $\mathcal{P}_N f$ mapped to the interval $[-1, 1]$. We define the *pseudo-spectral derivative*

to be the derivative of $\mathcal{P}_N f$ defined as

$$\mathcal{D}_N f = (\mathcal{P}_N f)' \tag{3.11}$$

The associated error is of the exponential type; namely, it depends on the smoothness of the original function f [99]. Thus, similar to the definition of the original interpolating polynomial, the derivative can be defined as

$$\mathcal{D}_N f(\xi_N) = \sum_{i=1}^{N} f(\xi_i)\lambda_i'(\xi) \tag{3.12}$$

The derivative evaluated at the interpolation nodes can be determined if we know the values of f (which can be calculated directly) and the derivative of the weight functions, λ_i'. These values can be calculated a priori and stored in the *pseudo-spectral differentiation matrix* D where $D_{ij} = \lambda_j'(\xi_i)$ for $i, j = 1, \ldots, N$. Finally knowing the differential matrix allows us to define the derivative of a function spectrally as

$$f_N' = D_N f_N \tag{3.13}$$

This is a general definition where the differentiation matrix is unique for the specific collocation method employed. Furthermore, the differentiation matrix may be uniquely defined for even or odd N. Such is the case for Fourier spectral methods. Since such restrictions are not necessary for the Chebyshev collocation methods utilized here we will follow the procedure given by Trefethen [95] and examine the case for $N = 2$ and $N = 3$ in an effort to extend our observations to a general case.

Consider the case of $N = 2$. The collocation nodes are $\xi_1 = 1$ and $\xi_2 = -1$ corresponding to the data points f_1 and f_2. Applying the definitions given by Eq. (3.8) and Eq. (3.9) we

determine that

$$\mathcal{P}_N f(\xi) = \frac{1}{2}(1+\xi)f_1 + \frac{1}{2}(1-\xi)f_2 \tag{3.14}$$

Algorithm B.2.1 on page 295 can be used to generate the Nth order interpolating polynomial automatically. Taking the derivative gives

$$\mathcal{D}_N f(\xi) = \frac{1}{2}f_1 - \frac{1}{2}f_2 \tag{3.15}$$

This result shows that the first row contains the coefficients of the polynomial for the first collocation point, ξ_1, and the second row for the second point. Therefore, from $\mathcal{D}_N f(\xi)$ at $\xi_i = \{-1, 1\}$, we determine the 2×2 pseudo-spectral differentiation matrix [95] to be

$$D_2 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \tag{3.16}$$

Now consider the case of $N = 3$. The collocation points are $\xi_1 = 1$, $\xi_2 = 0$, and $\xi_3 = -1$ and the interpolant is

$$\begin{aligned} \mathcal{P}_N f(\xi) &= \tfrac{1}{2}\xi(1+\xi)f_1 + (1+\xi)(1-\xi)f_2 + \tfrac{1}{2}\xi(\xi-1)f_3 \\ \mathcal{D}_N f(\xi) &= \left(\xi + \tfrac{1}{2}\right) f_1 - 2\xi f_2 + \left(\xi - \tfrac{1}{2}\right) f_3 \end{aligned}$$

In this instance, the differentiation matrix [95] is defined as

$$D_3 = \begin{bmatrix} \frac{3}{2} & -2 & \frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 2 & -\frac{3}{2} \end{bmatrix} \tag{3.17}$$

Again, the ith row of this 3×3 matrix is built by calculating the coefficients of the derivative polynomial for the ith collocation points ($\xi_i = 1$, 0, and -1).

The article by Fornberg discusses the connection between the Chebyshev spectral matrix and the coefficients of several common discretization schemes such as the centered 3-point finite difference discretization as well as the coefficients for the second-order Adams-Bashforth formula for the numerical solution of ODEs [100]. Higher-order spectral differentiation matrices are related to higher order finite difference formulas; however, since these are defined on uneven grids, they are not popular in standard applications [95].

From these two examples we form a basis for generalizing the Chebyshev pseudo-spectral differentiation matrix. The general formula may have first appeared in the book by Voight and coworkers [101].

Theorem 3.3.1. Chebyshev Differentiation Matrix *For each $N \geq 1$, let the rows and columns of the $(N+1) \times (N+1)$ Chebyshev spectral differentiation matrix D_N be indexed from 0 to N. The entries of this matrix are:*

$$D_N = \begin{cases} (D_N)_{11} = \dfrac{2(N-1)^2+1}{6}; & \\ (D_N)_{ii} = \dfrac{-\xi_i}{2\left(1-\xi_i^2\right)}; & i = 2, \ldots, N-1 \\ (D_N)_{ij} = \dfrac{d_i}{d_j}\dfrac{(-1)^{i+j}}{(\xi_i - \xi_j)}; & i \neq j, \quad i, j = 2, \ldots, N-1 \\ (D_N)_{NN} = -\dfrac{2(N-1)^2+1}{6}; & \end{cases} \tag{3.18}$$

The matrix element formulas can be more clearly understood graphically

$$D_N = \left[\begin{array}{c|ccc|c} \frac{2(N-1)^2+1}{6} & \cdots & \frac{2(-1)^{1+j}}{1-\xi_j} & \cdots & \frac{1}{2}(-1)^{1+N} \\ \hline \vdots & \ddots & \cdots & \frac{(-1)^{i+j}}{\xi_i-\xi_j} & \vdots \\ \frac{1}{2}\frac{(-1)^{i+1}}{\xi_i-1} & \vdots & \frac{-\xi_j}{2(1-\xi_j^2)} & \vdots & \frac{1}{2}\frac{(-1)^{i+N}}{\xi_i+1} \\ \vdots & \frac{(-1)^{i+j}}{\xi_i-\xi_j} & \cdots & \ddots & \vdots \\ \hline -\frac{1}{2}(-1)^{N+1} & \cdots & -\frac{2(-1)^{N+j}}{1+\xi_j} & \cdots & -\frac{2(N-1)^2+1}{6} \end{array}\right] \tag{3.19}$$

Higher order derivatives can be represented spectrally by simply raising the first order pseudo-spectral differentiation matrix to the corresponding power:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\xi} &\longrightarrow D_N \\ \frac{\mathrm{d}^2}{\mathrm{d}\xi^2} &\longrightarrow (D_N)^2 = D_N \times D_N \\ &\vdots \\ \frac{\mathrm{d}^n}{\mathrm{d}\xi^n} &\longrightarrow (D_N)^n = D_N \times D_N \times \ldots \times D_N \ n \text{ times} \end{aligned}$$

Computationally, this operation costs $\mathcal{O}(N^{3n})$ flops. Higher order spectral derivatives can also be computed via formula [102] or recurrence relations [103, 104] at a cost of only $\mathcal{O}(N^{2n})$. Since this operation needs to be computed only once and stored for later use, the former operation is generally sufficient.

Furthermore, the collection of spectral differentiation and integration codes detailed in the article by Weideman and Reddy [97] has become an invaluable tool for those undertaking spectral analysis in MATLAB. Their differentiation matrix suite is hosted by MATLAB

CENTRAL at http://www.mathworks.com/matlabcentral/fileexchange/29. The function `chebdif` generates N collocation points and the nth order pseudo-spectral Chebyshev differentiation matrix according to the syntax `[xi,D]=chebdif(N,n)` where `xi` is a vector defining the collocation points and `D(:,:,n)` is the $N \times N$, n^{th} order differentiation matrix. A copy of this can be found in Alg. B.3.1 on page 296. The following example illustrates the application of spectral differentiation matrices in finding the derivative of $\sin(x)$ over the interval $[0, 2\pi]$.

Example: Computing a Spectral Derivative

Since the Cheybshev polynomials and their spectral differentiation matrices are defined over the interval $[-1, 1]$, we must determine the proper transformation to map the solution over the domain $[0, 2\pi]$. We have

$$\xi = \frac{x}{\pi} - 1 \qquad \text{so} \qquad \frac{\mathrm{d}}{\mathrm{d}x} = \frac{1}{\pi}\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{3.20}$$

First, we determine the $N \times N$, 1st order spectral differentiation matrix

```
N=20;
[xi,D]=chebdif(N,1);
```

Since this is on the interval $[-1, 1]$, we must apply our transform to both the collocation points and the differentiation matrix:

```
x=pi*(xi+1);d_dx=1/pi*D(:,:,1);
```

: refer to Eq. (3.20)

Lastly, we plot the results compared to several values of N in Fig. 3.4. Clearly the $N = 5$ case is insufficient; however, the solution resolves itself well for $N = 10$ and $N = 20$.

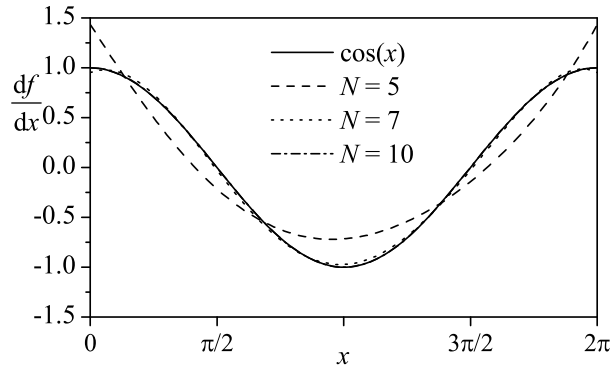


Figure 3.4: Spectral differentiation of $\sin(x)$ for three polynomial orders plotted against the exact derivative.

```
plot(x,d_dx*sin(x))
```

For $N = 10$, the spectral solution is indistinguishable from the exact. Algorithm B.5.1 computes spectral derivatives and plots the interpolating polynomial.

3.4 Solving ODEs with Chebyshev Collocation

We have successfully formulated all the tools required to solve equations of the form

$$
\begin{aligned}
&Q_0(x)f^{(n)}(x) + Q_1(x)f^{(n-1)}(x) + \ldots + Q_{n-1}(x)f'(x) + Q_n(x)f(x) = g(x) \\
&\text{with} \quad \begin{cases} K_0(A)f^{(n)}(A) + K_1(A)f^{(n-1)}(A) + \ldots + K_{n-1}(A)f'(A) + K_n(A)f(A) = a; \\ H_0(B)f^{(n)}(B) + H_1(B)f^{(n-1)}(B) + \ldots + H_{n-1}(B)f'(A) + H_n(B)f(B) = b; \\ \quad \vdots \end{cases}
\end{aligned}
\tag{3.21}
$$

where the number of boundary conditions is adjusted to define a well-posed problem.

To solve equations of this type we must first express it spectrally. This is done by creating an operator matrix acting on the function, $f(x)$. This is not as daunting as it seems. In fact, we already have all the tools we need, just not the method. First, we write

Eq. (3.21) in the operator form

$$\left[Q_0(x)\frac{\mathrm{d}^{(n)}}{\mathrm{d}x^{(n)}} + Q_1(x)\frac{\mathrm{d}^{(n-1)}}{\mathrm{d}x^{(n-1)}} + \ldots + Q_{n-1}(x)\frac{\mathrm{d}}{\mathrm{d}x} + Q_n(x)\right] f(x) = g(x)$$

$$\text{with} \quad \begin{cases} \left[K_0(A)\frac{\mathrm{d}^{(n)}}{\mathrm{d}x^{(n)}} + K_1(A)\frac{\mathrm{d}^{(n-1)}}{\mathrm{d}x^{(n-1)}} + \ldots + K_{n-1}(A)\frac{\mathrm{d}}{\mathrm{d}x} + K_n(A)\right] f(A) = a; \\ \left[H_0(B)\frac{\mathrm{d}^{(n)}}{\mathrm{d}x^{(n)}} + H_1(B)\frac{\mathrm{d}^{(n-1)}}{\mathrm{d}x^{(n-1)}} + \ldots + H_{n-1}(B)\frac{\mathrm{d}}{\mathrm{d}x} + H_n(B)\right] f(B) = b; \\ \quad \vdots \end{cases} \tag{3.22}$$

Just as with spectral differentiation, the first step is to determine the appropriate transformation to map the solution over the interval $[-1, 1]$. We can formulate a general equation to map any interval appropriately. It is,

$$x = \frac{B}{2}(\xi + 1) - \frac{A}{2}(\xi - 1) \quad \longleftrightarrow \quad \xi = \frac{2x - (B + A)}{B - A}$$

$$\text{likewise,} \qquad \frac{\mathrm{d}}{\mathrm{d}x} = \frac{2}{B - A}\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{3.23}$$

Now it becomes simply a task of expressing the operator spectrally. Therefore, $x \to x_i = \{x_N, x_{N-1}, \ldots, x_1\}$, $f(x) \to f_i = \{f(x_N), f(x_{N-1}), \ldots, f(x_1)\}$, and $g(x) \to g_i = \{g(x_N), g(x_{N-1}), \ldots, g(x_1)\}$. Be aware that the function `chebdif` returns collocation points from right to left. In other words, x_1 corresponds to the right bound and x_N to the left. It is important to keep this in mind when applying boundary conditions. The coefficients form the diagonal matrix, $(Q_j)_{ii}$. Any general function $q_j(x)$ will fill the matrix, $(Q_j)_{ii}$ according to

$$(Q_j)_{ii} = \begin{bmatrix} q_j(x_N) & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & q_j(x_1) \end{bmatrix} \tag{3.24}$$

The derivatives become pseudo-spectral differentiation matrices directly such that the general differential equation is written spectrally as

$$\left[(Q_0)_{ii}(\bar{D}_N)^n + (Q_1)_{ii}(\bar{D}_N)^{n-1} + \ldots + (Q_{n-1})_{ii}\bar{D}_N + (Q_n)I\right] f_i = g_i \tag{3.25}$$

By collecting terms we can write the governing equation in matrix form as

$$A_{ij} f_i = B_i \tag{3.26}$$

where A_{ij} is the operator matrix and B_i is the vector containing the forcing function.

Now the question remains: how do we handle boundary conditions? If we think in terms of polynomial interpolation, this becomes a fairly easy concept to understand. Recall at each collocation point the interpolating polynomial will exactly match the function. Since the set of collocation points include the bounds, it is a matter of overwriting the first and last line of the operator matrix and forcing function with the appropriate boundary conditions at the right and left, respectively. This way we still have N equations and N unknowns that ensure that the interpolating polynomial will match the boundary conditions exactly. In retrospect, there are a few things to consider:

1. If the ODE is a boundary value problem (Dirichlet conditions), then we must overwrite the 1st and N^{th} rows of A and B with the 1st and N^{th} rows of the identity matrix. For example

$$\begin{cases} f(A) = a \\ f(B) = b \end{cases} \longrightarrow \begin{cases} [I_{N,:}]f_N = a \\ [I_{1,:}]f_1 = b \end{cases} \tag{3.27}$$

2. If the boundary conditions are Neumann type (derivative conditions), then we use the spectral differentiation matrix instead of the identity matrix. For example

$$\begin{cases} f'(A) = a \\ f'(B) = b \end{cases} \longrightarrow \begin{cases} [\bar{D}_{N,:}]f_N = a \\ [\bar{D}_{1,:}]f_1 = b \end{cases} \tag{3.28}$$

The use the overbar to denote the pseudo-spectral differentiation matrix mapped over arbitrary domain $[A, B]$, $\bar{D}$, and reserve D to preserve the differentiation matrix over the Chebyshev space, $[-1, 1]$.

3. If the problem is an initial value problem, then we still must overwrite the 1st row in the operator matrix corresponding to the right bound because defining any value at that collocation point, even through the unmodified operator matrix, renders an overdetermined system of equations. Thus,

$$\begin{cases} f(A) = a \\ f'(A) = b \end{cases} \longrightarrow \begin{cases} [I_{N,:}]f_N = a \\ [\bar{D}_{N,:}]f_N = b \end{cases} \tag{3.29}$$

As we see in all three considerations, we never use the 1st or N^{th} rows of the A_{ij} or B_i. We could define $\tilde{A}_{ij} = A_{2:N-1}$ as others have, but this is not necessary. As an example, the boundary conditions for a boundary value problem will be written as

$$\begin{cases} \left[(K_0)_{N,:}(\bar{D}_N)^n_{N,:} + (K_1)_{N,:}(\bar{D}_N)^{n-1}_{N,:} + \ldots + (K_{n-1})_{N,:}(\bar{D}_N)_{N,:} + (K_n)_{N,:}I_{N,:}\right] f_N = a; \\ \left[(H_0)_{1,:}(\bar{D}_N)^n_{1,:} + (H_1)_{1,:}(\bar{D}_N)^{n-1}_{1,:} + \ldots + (H_{n-1})_{1,:}(\bar{D}_N)_{1,:} + (H_n)_{1,:}I_{1,:}\right] f_1 = b; \end{cases} \tag{3.30}$$

These examples must be extended to higher order derivative boundary conditions accordingly.

3.4.1 Example: A First Order ODE with Variable Coefficients

There is one more caveat to the statements above. Clearly, for a first order ODE we only have one boundary condition therefore when we solve a first order differential equation with spectral methods, we are unable to replace both the 1st and N^{th} rows with boundary condition equations. Rather, we only replace the row corresponding to the boundary in which the condition is specified. The other row remains unmodified. Consider the first order ODE

$$f'(x) + \cos(4x)f(x) = 0; \quad f(0) = 1 \tag{3.31}$$

with the exact solution

$$f(x) = e^{-\frac{1}{4}\sin(4x)} \tag{3.32}$$

Referring to Eq. (3.23), we can determine the appropriate map over the interval $[-1, 1]$. It is,

$$x = \frac{\pi}{2}(\xi + 1) \quad \text{and} \quad \frac{\mathrm{d}}{\mathrm{d}x} = \frac{2}{\pi}\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{3.33}$$

Now we define the number of collocation points, the left and right bounds A, B, respectively, the value at the bounds a, and calculate the differentiation matrix.

```
N=10;
A=0;B=pi;a=1;
[xi,D]=chebdif(N,1);
```

Next, we apply the transformation to map the collocation points and the derivative matrix to the appropriate domain.

```
x=pi/2*(xi+1);
```

```
d_dx=2/pi*D(:,:,1);
```

: refer to Eq. (3.33)

Since the right hand side is zero we define `Bvec=zeros(N,1)`. Next define the operator matrix, A.

```
Amat=d_dx+diag(cos(4*x))*I;
```

: refer to Eq. (3.31)

The boundary condition is on the left bound so we have only

```
Amat(N,:)=I(N,:);
Bvec(N)=a;
```

: refer to Eq. (3.31)

Since there is only one boundary condition, we must let the opposite bound remain unspecified.

Finally, we solve the system using `f=Amat\Bvec`. The results are plotted in Fig. 3.5. This equation requires a higher resolution than some to return accurate results. This is due to its oscillatory solution. Finding the optimum number N becomes an iterative process and left up to the researcher. In general, the collocation grid must be fine enough to contain two points per wavelength [95]. The complete algorithm is found in Alg. B.6.1 on page 300. The error is less than 5×10^{-4} for $N = 20$.

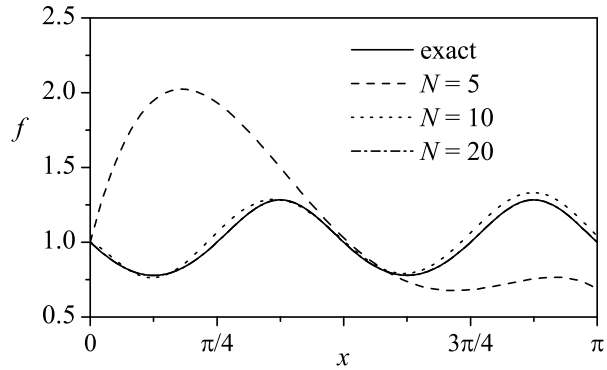


Figure 3.5: Spectral solution of Eq. (3.31) for three polynomial orders plotted against the exact solution.

3.4.2 Example: A Second Order BVP with Variable Coefficients

Consider the boundary value problem

$$f''(x) + xf'(x) + f(x) = x \quad \text{with} \quad \begin{cases} f(0) = 0 \\ f(4) = 1 \end{cases} \tag{3.34}$$

The first step is to determine the appropriate transformation to map the solution over the interval $[-1, 1]$. Referring to Eq. (3.23), we can determine the appropriate map, namely,

$$x = 2(\xi + 1) \quad \text{and} \quad \frac{\mathrm{d}}{\mathrm{d}x} = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{3.35}$$

Now within the code we first define the number of collocation points and calculate the differentiation matrix.

```
N=10;
[xi,D]=chebdif(N,1);
```

Next, we apply the transformation to map the collocation points and the derivative matrix to the appropriate domain:

```
x=2*(xi+1);
d_dx=1/2*D(:,:,1);
d2_dx2=d_dx^2;
```

: refer to Eq. (3.35)

The higher order derivatives can be calculated directly by the function `chebdif`; however, the coordinate transformation must be calculated for each higher order derivative individually. By simply applying the transformation to the first order derivative and raising it to the appropriate power (in this case, 2), the transformation is propagated appropriately and automatically.

Now we discretize any variable coefficients and forcing functions. Also, for clarity, we will define the identity matrix as well:

```
Q=diag(x);
Bvec=x;
I=eye(N);
```

: refer to Eq. (3.34)

The vector B contains the values of the forcing function evaluated at the collocation points. Similarly, the first entry corresponds to the right bound and the last to the left. At this point we can define the operator matrix, A. It will be the sum of the second derivative spectral matrix, the first derivative multiplied from the left by the coefficient matrix, and the identity matrix. We have

```
Amat=d2_dx2+Q*d_dx+I;
```

: refer to Eq. (3.34)

Recalling that spectral methods work by setting up and solving a system of N equation with N unknowns, we can take the boundary conditions as equations and add them to the operator and forcing function matrices directly. The right boundary corresponds to the first row of entries and the left to the last so that we can impose these conditions by overwriting the first and last row in A and B with the right and left boundary condition operators and values, respectively. We do this with the commands

```
Amat(1,:)=I(1,:);Bvec(1)=1;
Amat(N,:)=I(N,:);Bvec(N)=0;
```

: refer to Eq. (3.34)

Lastly, we solve for the solution vector by using the MATLAB operation `f=Amat\Bvec`. The algorithm to solve this equation is supplied in the appendix as Alg. B.7.1 on page 301. The results are plotted against the exact solution in Fig. 3.6, specifically

$$f(x) = \frac{x\text{erfi}\left(2\sqrt{2}\right) - 2e^{8-\frac{x^2}{2}}\text{erfi}\left(\frac{x}{\sqrt{2}}\right)}{2\text{erfi}\left(2\sqrt{2}\right)} = \frac{x}{2} - \frac{D_+\left(\frac{x}{\sqrt{2}}\right)}{D_+\left(2\sqrt{2}\right)} \tag{3.36}$$

where D_+ is the Dawson function defined as

$$D_+ = e^{-x^2}\int_0^x e^{t^2}dt = \frac{\sqrt{\pi}}{2}e^{-x^2}\text{erfi}(x) \tag{3.37}$$

The the maximum local error, defined by $||f(x) - F(x)||_\infty$, is found to be less than 5×10^{-4} for $N = 10$. Another source of error comes from numerically solving the system $A_{ij}f_i = B_i$. This source is not significant in this example but does become important for solution matrices requiring a large number of collocation points, systems of ODEs, and any other operator matrix that is singular or nearly singular. This error can be quantified by calculating $||Af - B||_\infty$.

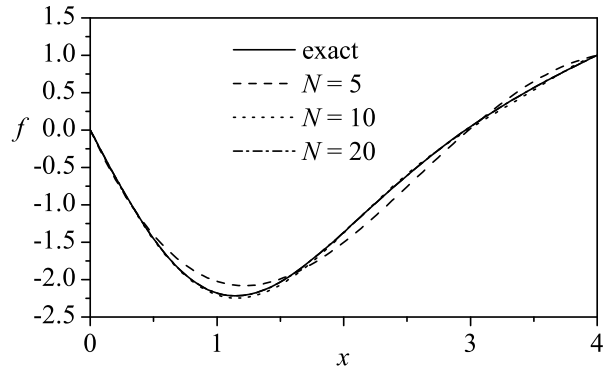


Figure 3.6: Spectral solution of Eq. (3.34) for three polynomial orders plotted against the exact solution.

3.4.3 Example: A System of Differential Equations

Many facets of research cannot be solved with a single differential equation alone. Often we are faced with a linear system of differential equations that must be solved simultaneously. Such is the case with classic one-dimensional hydrodynamic stability formulations. In many cases Runge-Kutta solvers will quickly integrate the equations, but shed little light on their spectral characteristics (eigenvalues) without requiring further calculations. For this type of analysis, we must once again resort to collocation methods. While solving single differential equations is relatively straightforward, systems of differential equations add a significant level of difficulty in the numeric formulation. For this reason we will carefully detail the solution to the system

$$\begin{cases} f''(x) + xf'(x) - g(x) = \sin(4x) \\ g''(x) + g'(x) + f(x) = \cos(4x) \end{cases} \quad \text{with} \quad \begin{cases} f(0) = 1; \quad f'(0) = 10 \\ g(0) = 1; \quad g'(0) = 0 \end{cases} \tag{3.38}$$

First, we need to identify the form of the coefficient matrix. We start as before and rewrite the equation in operator form

$$\begin{cases} \left(\dfrac{\mathrm{d}^2}{\mathrm{d}x^2} + x\dfrac{\mathrm{d}}{\mathrm{d}x}\right) f(x) & + & (-1)g(x) & = & \sin(4x) \\ (1)f(x) & + & \left(\dfrac{\mathrm{d}^2}{\mathrm{d}x^2} + \dfrac{\mathrm{d}}{\mathrm{d}x}\right) g(x) & = & \cos(4x) \end{cases}$$

$$\text{with BCs} \quad \begin{cases} (1)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) f(0) & + & (0)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) g(0) & = & 10 \\ (1)f(0) & + & (0)g(0) & = & 1 \\ (0)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) f(0) & + & (1)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) g(0) & = & 0 \\ (0)f(0) & + & (1)g(0) & = & 1 \end{cases}$$

While this may be an awkward way of writing the equations, especially the boundary conditions, it provides a natural step from the original system to its equivalent spectral form. From here we can write

$$\begin{bmatrix} \left(\dfrac{\mathrm{d}^2}{\mathrm{d}x^2} + x\dfrac{\mathrm{d}}{\mathrm{d}x}\right) & (-1) \\ (1) & \left(\dfrac{\mathrm{d}^2}{\mathrm{d}x^2} + \dfrac{\mathrm{d}}{\mathrm{d}x}\right) \end{bmatrix} \begin{bmatrix} f(x) \\ g(x) \end{bmatrix} = \begin{bmatrix} \sin(4x) \\ \cos(4x) \end{bmatrix} \tag{3.39}$$

and

$$\begin{bmatrix} (1)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) & (0)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) \\ (1) & (0) \\ (0)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) & (1)\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) \\ (0) & (1) \end{bmatrix} \begin{bmatrix} f(0) \\ g(0) \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{3.40}$$

Before we proceed with these matrices, lets first consider the transform required to map the domain to $[-1, 1]$. Based on Eq. (3.23) we find

$$x = 5(\xi + 1); \quad \frac{\mathrm{d}}{\mathrm{d}x} = \frac{1}{5}\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{3.41}$$

The operator matrices may now be written in spectral terms according to the outlined procedure. The system becomes

$$\begin{bmatrix} (\bar{D}_N)^2 + x_{ii}\bar{D}_N & (-I_N) \\ (I_N) & (\bar{D}_N)^2 + \bar{D}_N \end{bmatrix} \begin{bmatrix} f(x_i) \\ g(x_i) \end{bmatrix} = \begin{bmatrix} \sin(4x_i) \\ \cos(4x_i) \end{bmatrix} \tag{3.42}$$

where $\bar{D}_N$ denotes the Chebyshev spectral differentiation matrix mapped to the original x domain. Also recall that the subscripts on the coefficient x_{ii} define the coefficient as a diagonal matrix

$$x_{ii} = \begin{bmatrix} x_N & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_1 \end{bmatrix} \tag{3.43}$$

and the forcing functions are column vectors.

In MATLAB we implement this automatically with the code

```
N=20;
i=1:1:N;j=1:1:N;
A=0;B=10;
[xi,D]=chebdif(N,1);
x=B/2*(xi+1)-A/2*(xi-1);
d_dx=2/(B-A)*D(:,:,1);
d2_dx2=d_dx^2;
```

```
I=eye(N);
Bvec(i)=sin(4*x);
Bvec(N+i)=cos(4*x);
Amat(i,j)=d2_dx2+diag(x)*d_dx;
Amat(i,N+j)=-I;
Amat(N+i,j)=I;
Amat(N+i,N+j)=d2_dx2+d_dx;
```

Remember that computation for a second order equation occurs over the inner nodes only with the first and last row being explicitly defined by the boundary conditions. We can rewrite our matrix to reflect this criterion as

$$\begin{bmatrix} (\bar{D}_N)_{N,:} & (0)(\bar{D}_N)_{N,:} \\ (\bar{D}_N)^2_{2:N-1,1,:} + (x_{ii})_{2:N-1,1,:}(\bar{D}_N)_{2:N-1,1,:} & (-I_N)_{2:N-1,1,:} \\ (I_N)_{N,:} & (0)(I_N)_{N,:} \\ (0)(\bar{D}_N)_{N,:} & (\bar{D}_N)_{N,:} \\ (I)_{2:N-1,1,:} & (\bar{D}_N)^2_{2:N-1,1,:} + (\bar{D}_N)_{2:N-1,1,:} \\ (0)(I_N)_{N,:} & (I_N)_{N,:} \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_{2:N-1}) \\ f(x_N) \\ g(x_1) \\ g(x_{2:N-1}) \\ g(x_N) \end{bmatrix} = \begin{bmatrix} 10 \\ \sin(4x_{2:N-1}) \\ 1 \\ 0 \\ \cos(4x_{2:N-1}) \\ 1 \end{bmatrix} \tag{3.44}$$

While Eq. (3.44) is not cast in proper matrix form, it offers a visual representation of what will be required for coding. To clarify, the first column refers to the function $f(x)$ and the second to $g(x)$. Also, the first three rows are from the first equation and the 4th through 6th come from the second.

Now it becomes more evident why we originally stated the boundary conditions in such an irregular way. In order to spectrally apply boundary conditions, they must span the entire row of the operator matrix. This is easy to implement for a single equation but for a system each boundary condition must be thought to have information for each function in the solution (even if the corresponding contribution is zero). Most of the time a boundary condition is only specified for one function at a time so any other function has a zero coefficient. Thus, to ensure the derivative condition in the first row only acts on $f(x)$, the elements corresponding to $g(x)$ must be zeroed. Also remember that we take the Nth row of the derivative and identity matrices since that corresponds to the left side of the domain. The boundary conditions are coded as

```
Amat(1,j)=d_dx(N,:);     % Define the IV operator on u from BC on u
Amat(1,N+j)=0;           % Define the IV operator on y from BC on u
Bvec(1)=10;              % Define the IV on u
Amat(N,j)=I(N,:);        % Define the BC operator on u from BC on u
Amat(N,N+j)=0;           % Define the BC operator on y from BC on u
Bvec(N)=1;               % Define the BC on u
Amat(N+1,j)=0;           % Define the IV operator on u from BC on y
Amat(N+1,N+j)=d_dx(N,:);  % Define the IV operator on y from BC on y
Bvec(N+1)=0;             % Define the IV on y
Amat(2*N,j)=0;           % Define the BC operator on u from BC on y
Amat(2*N,N+j)=I(N,:);    % Define the BC operator on y from BC on y
Bvec(2*N)=1;             % Define the BC on y
```

Clearly, this takes great care to position the conditions properly. The complete code is found in Alg. B.8.1 on page 303. Results are plotted against a 1000 step Runge-Kutta solution in Fig. 3.7. This system requires approximately 30 collocation points to compute a solution whose disparity to the 1000 step Runge-Kutta solution is less than 5×10^{-4}.

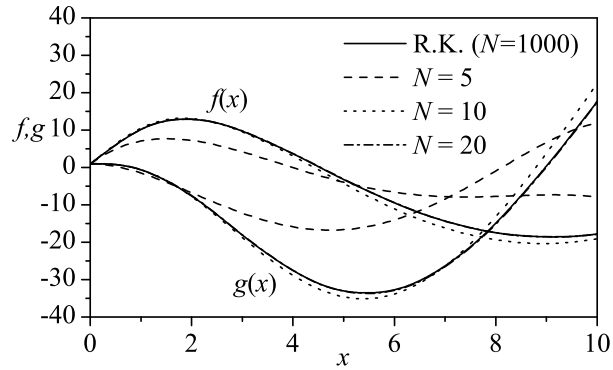


Figure 3.7: The solution to the system of ODEs given in Eq. (3.38) for three collocation numbers.

3.5 Eigenvalue Problems with ODEs

A very important part of hydrodynamic instability research, and many other areas of science, is the generalized eigenvalue problem of the form

$$A_{ij} f_i = \lambda B_{ij} f_i \tag{3.45}$$

where A_{ij} and B_{ij} (as opposed to vector B_i used above) are operator matrices, λ is an eigenvalue, and f_i is the eigenvector. The expression $A_{ij} - \lambda B_{ij}$ is called the matrix pencil. If B_{ij} is the identity matrix, then the problem becomes $A_{ij} - \lambda I_N$; which is the eigenvalue problem for a single matrix. Single matrix eigenvalue problems are certainly important in their own way. Typically the single matrix consists of the coefficients of a simultaneous system of algebraic or differential equations. In these cases the eigenvalues appear explicitly and are found by setting the Wronskian equal to zero.

Generalized eigenvalue problems appear when the eigenvalue itself appears in the governing equation or system and multiplies an operator acting on the solution function. A simple example of this equation for an undamped oscillator equation, $\ddot{f} + \omega^2 x^2 f = 0$, where $\lambda = \omega^2$ and correspond to the zeros of the solution. In instability research, the eigenvalues relate to both the frequency and growth rate of oscillations.

For the single matrix eigenvalue problem, the eigenvalues are solved from the equation $\det(A_{ij} - \lambda I_N)$. If B_{ij} is nonsingular, then the generalized eigenvalue problem can be reduced to a single matrix eigenvalue problem by

$$A_{ij}B_{ij}^{-1} - \lambda B_{ij}B_{ij}^{-1} = D_{ij} - \lambda I_N = 0; \quad D = A_{ij}B_{ij}^{-1} \tag{3.46}$$

To emphasize, B_{ij} clearly cannot be singular. For practical purposes though, this strategy is ill-advised. For large matrices (actually any matrix greater than 2×2), calculating an inverse analytically becomes daunting. Calculating an inverse numerically introduces significant errors that are amplified in the calculation of the spectrum. Therefore it becomes important to employ similarity transformation matrices that reduce both matrix A_{ij} and B_{ij} to triangular form where the eigenvalues appear as the diagonal elements. This is typically done by way of the QZ* [68, 69] (or LZ [70, 71]) for dense matrices or Arnoldi algorithm [105–108] for sparse. MATLAB implements the QZ and Arnoldi solvers with the commands `eig(A,B)` and `eigs(A,B)`, respectively. We will discuss the details of these methods in Ch. 4.

3.5.1 Example: Eigenvalues of the Bessel Equation

Here, we will determine the spectral collocation and solution of the Bessel equation

$$x^2 f''(x) + x f'(x) + \mu^2 x^2 f(x) = 0 \quad \text{with} \quad \begin{cases} f'(0) = 0 \\ f(L) = 0 \end{cases} \tag{3.47}$$

The boundary conditions do not admit a clear solution; rather, they imply the condition $J_0(\mu L) = 0$ so that the eigenvalue μ must correspond to the zeros of the Bessel function.

*A generalized eigensolver using similarity transformations to reduce A_{ij} and B_{ij} to upper triangular where the generalized eigenvalues are known to be a_{ii}/b_{ii}.

Since the equation contains the eigenvalue, we must segregate terms like

$$\left(x^2\frac{\mathrm{d}^2}{\mathrm{d}x^2}+x\frac{\mathrm{d}}{\mathrm{d}x}\right)f(x)+\lambda\left(x^2\right)f(x)=0 \tag{3.48}$$

where $\lambda=\mu^2$. If the eigenvalue appears in the boundary conditions, then we must correctly segregate them as well. Now we write the equations in the spectral matrix form given by $A_{ij}f_i=\lambda B_{ij}f_i$ with

$$A_{ij}=x_{ii}^2(\bar{D}_N)^2+x_{ii}(\bar{D}_N)\quad\text{and}\quad B_{ij}=x_{ii}^2 \tag{3.49}$$

This is implemented with the code

```
Amat=diag(x.^2)*d2_dx2+diag(x)*d_dx;
Bmat=diag(x.^2);
```

: refer to Eq. (3.49)

Had a forcing function been present we can write it as well but it will not appear in the generalized eigenvalue equation. To understand this, consider the forced oscillator equation

$$\ddot{f}+c\dot{f}+\omega^2 f=\cos(\Omega t) \tag{3.50}$$

The homogenous solution is found first and with it the eigenvalue, ω, dictating the unperturbed natural frequency. The forcing function and the frequency Ω persist after the transient period has expired but appear only as an addition to the overall spectrum of the homogenous equation. Figure 3.8 distinctly identifies the frequency Ω as an addition to the spectrum dictated by ω. The same is true for the value of the boundary conditions: Only their operators are included in the matrices.

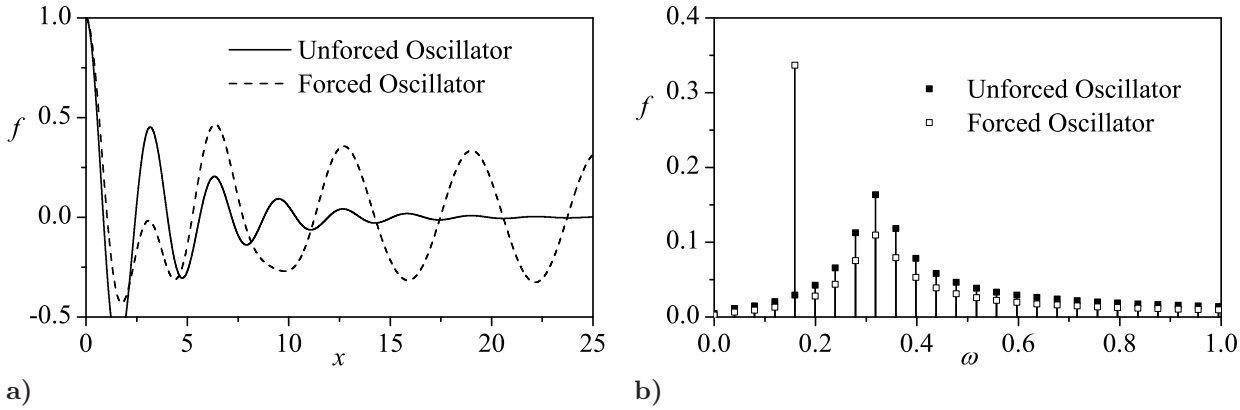


Figure 3.8: Unforced and sinusoidally forced damped oscillator for $\omega = 2$, $\Omega = 1$, and $c = \frac{1}{2}$ (3.8a) and corresponding FFT showing the damped natural frequency $\sqrt{\omega^2 - c^2}/2\pi \approx 0.308$ and the forcing frequency $\Omega/2\pi \approx 0.159$ (3.8b).

The boundary conditions are applied according to

$$\begin{cases} \left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) f(0) + \lambda\,(0) \left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right) f(0) = 0 \\ f(L) + \lambda(0) f(L) = 0 \end{cases} \longrightarrow \begin{cases} (\bar{D}_N)_{N,:} f(0) + \lambda\,(0)\,(\bar{D}_N)_{N,:} f(0) = 0 \\ (I_N)_{1,:} f(L) + \lambda(0)(I_N)_{1,:} f(L) = 0 \end{cases} \tag{3.51}$$

The corresponding code looks like

```
Amat(1,:)=I(1,:);
Amat(N,:)=d_dx(N,:);
Bmat(1,:)=0;
Bmat(N,:)=0;
```

Finally we solve for the eigenvalues and eigenvectors using the built-in command `[V,Lam]=`**`eig`**`(A,-B)` where the command **`eig`**`(A,-B)` computes the eigenvectors, `V`, and the eigenvalues per interval length, `Lam`. The B matrix is multiplied by negative one because it appears positive on the left-hand-side of the generalized eigenvalue equation where we

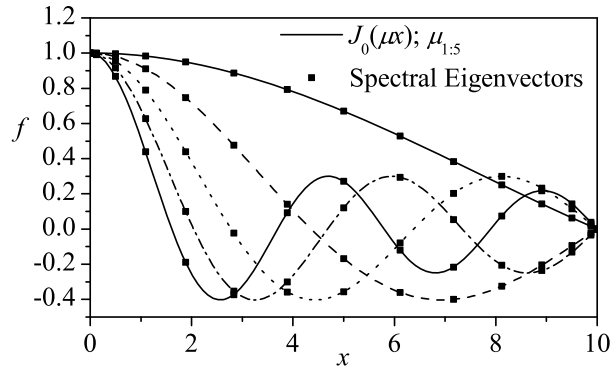


Figure 3.9: The eigenvectors plotted against the Bessel function of the first kind for the first five eigenvalues for $N = 15$ over the domain $0 \leq x \leq 10$.

have defined it as positive on the right. This is important to return the correct form of the equation to calculate the eigenvalues.

It must remain very clear that the code solves the spectrum for Eq. (3.45). Therefore, the output from the code is related to the original eigenvalue by $\mu = \sqrt{\lambda}$. Furthermore, the output automatically computes the eigenvalues for the problem defined over an arbitrary domain. Comparing eigenvalues over differing domains can only be done by normalizing over the domain. This is done by taking μL. These conversions are problem specific since the relationship between μ and λ may differ.

The complete code is found in Alg. B.9.1 on page 306. Table 3.2 shows the spectral values versus the accepted zeros of the Bessel function of the first kind. Two things to notice: 1) The spectral values contain two infinite eigenvalues and therefore only return results for $N - 2$ physical eigenvalues and 2) as a rule of thumb, approximately $N/3$ eigenvalues have a percent error less than 5×10^{-4}. This is in conjunction with the practical requirement of having at least two collocation nodes per wavelength [95]. We see excellent agreement between the spectral eigenvector and the Bessel function of the first kind for the first five eigenvalues in Fig. 3.9.

Table 3.2: Spectral roots of $J_0(\mu L)$ for $0 \leq x \leq L$ compared to their accepted values. $\%\ E = |\lambda_e - \mu|/\lambda_e$ where λ_e is the root of $J_0(\lambda_e)$ for $0 \leq x \leq 1$.

	μL							
$\lambda_e L$	$N = 5$	$\%\ E$	$N = 10$	$\%\ E$	$N = 15$	$\%\ E$	$N = 20$	$\%\ E$
2.4048	2.4040	0.0323	2.4048	1.305E-08	2.4048	3.342E-12	2.4048	1.455E-11
5.5201	5.6884	3.0493	5.5201	2.781E-05	5.5201	1.833E-10	5.5201	1.818E-12
8.6537	8.4665	2.1641	8.6539	0.0023	8.6537	1.187E-07	8.6537	4.516E-13
11.7915			11.7831	0.0712	11.7915	5.652E-06	11.7915	7.209E-10
14.9309			15.0997	1.1306	14.9309	1.428E-04	14.9309	3.385E-08
18.0711			17.4414	3.4842	18.0717	0.0036	18.0711	1.880E-06
21.2116			27.6179	30.2016	21.1873	0.1147	21.2116	7.371E-06
24.3525			36.4102	49.5133	24.5159	0.6713	24.3523	5.391E-04
27.4935					26.9492	1.9797	27.4952	0.0063
30.6346					33.7323	10.1116	30.6107	0.0781
33.7758					37.6674	11.5218	33.9231	0.4361
36.9171					64.9955	76.0579	36.3381	1.5683
40.0584					86.4115	115.7138	42.1466	5.2128
43.1998							45.2732	4.7995
46.3412							59.8774	29.2098
49.4826							67.4442	36.2988
52.6241							118.6801	125.5244
55.7655							158.2153	183.7154
58.9070								
62.0485								

3.6 Solving PDEs with Chebyshev Collocation

Our discussion on spectral methods for ordinary differential equations is sufficient to cover one-dimensional stability equations. For higher dimensional stability analysis, we are forced to breach the realm of *partial differential equations*. As expected, this will add significant difficulty to both the analytic formulation of the problem as well as the computational requirements. To aid us in the solution, we need to introduce some new concepts. A two-dimensional problem will correlate to a two-dimensional grid based on directionally independent Chebyshev points. Such a grid is called a "*tensor product grid*" [95]. This grid will be most dense at the boundaries and least dense at the middle.

To work with tensor product spectral grids we must know how to use *Kronecker products*. The Kronecker product of two matrices is denoted by $A_{ij} \otimes B_{mn} = C_{i\times m\ j\times n}$ where $C_{i\times m\ j\times n}$ is a block matrix such that each block is built as $a_{ij}B_{mn}$. For clarity the Kronecker product appears as:

$$A_{ij} \otimes B_{mn} = \begin{bmatrix} a_{11}B_{mn} & \cdots & a_{1j}B_{mn} \\ \vdots & \ddots & \vdots \\ a_{i1}B_{mn} & \cdots & a_{jj}B_{mn} \end{bmatrix} \tag{3.52}$$

so that each element of A_{ij} is multiplied by the entire matrix B_{mn}.

We can use the Kronecker product in spectral methods to build a single operator matrix from a two-dimensional collocation. Specifically derivatives with respect to one independent variable will take the form $(I_N) \otimes (D_N)^n$ and derivatives with respect to the other will be $(D_N)^n \otimes (I_N)$. We can compute this in MATLAB with the command `kron(X,Y)`. For example

the first derivative with respect to x with $N = 3$ can be given as

$$D_3^{\xi} = D_3 \otimes I_3 = \left[\begin{array}{ccc|ccc|ccc} 1.5 & & & -2.0 & & & 0.5 & & \\ & 1.5 & & & -2.0 & & & 0.5 & \\ & & 1.5 & & & -2.0 & & & 0.5 \\ \hline 0.5 & & & 0.0 & & & -0.5 & & \\ & 0.5 & & & 0.0 & & & -0.5 & \\ & & 0.5 & & & 0.0 & & & -0.5 \\ \hline -0.5 & & & 2.0 & & & -1.5 & & \\ & -0.5 & & & 2.0 & & & -1.5 & \\ & & -0.5 & & & 2.0 & & & -1.5 \end{array}\right] \tag{3.53}$$

and the first derivative with respect to y as

$$D_3^{\eta} = I_3 \otimes D_3 = \left[\begin{array}{ccc|ccc|ccc} 1.5 & -2.0 & 0.5 & & & & & & \\ 0.5 & 0.0 & -0.5 & & & & & & \\ -0.5 & 2.0 & -1.5 & & & & & & \\ \hline & & & 1.5 & -2.0 & 0.5 & & & \\ & & & 0.5 & 0.0 & -0.5 & & & \\ & & & -0.5 & 2.0 & -1.5 & & & \\ \hline & & & & & & 1.5 & -2.0 & 0.5 \\ & & & & & & 0.5 & 0.0 & -0.5 \\ & & & & & & -0.5 & 2.0 & -1.5 \end{array}\right] \tag{3.54}$$

These Kronecker product definitions of the partial derivatives in x and y can be switched as long as it is clear and consistent in the numerical solver. In fact, Trefethen defines the derivatives exactly opposite of that shown above [95]. However, the nature of the equation with its boundary conditions can require one definition versus another. In some cases a

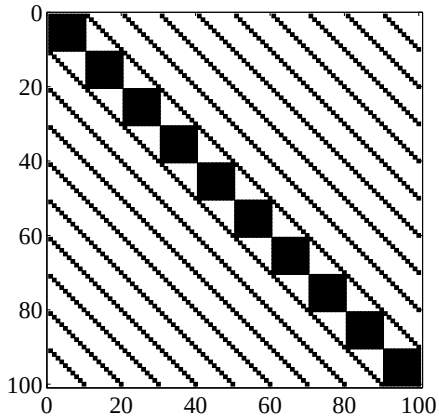
numerical oscillation can be observed in the direction of the derivative defined by $I_N \otimes D_N$. This oscillation can range from minor to significant. For consistency and accuracy we attempt to use the derivatives as defined above whenever possible. Regardless, these operations are indicative of the the Kronecker sum

$$A_{ij} \oplus B_{mn} = A_{ij} \otimes I_N + I_N \otimes B_{mn} \tag{3.55}$$

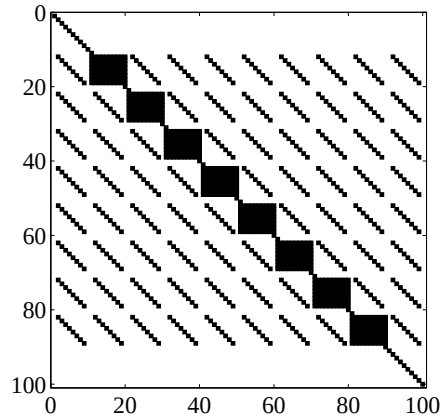
which will allow us to combine operator matrices from both directions so we can solve the spectral equation as usual. We see in Fig. 3.10, that these matrices are sparse but not so sparse that we must resort to special methods for singular or sparse matrices. Boundary conditions increase the sparsity but only around the outer perimeter. Finite element discretization would result in much sparser matrices and require hundreds or thousands of collocation points where as Chebyshev collocation and Kronecker products require tens to hundreds of points for accurate resolution of the solution [95].

By defining directional derivatives as Kronecker products, we can solve partial differential equations of the form

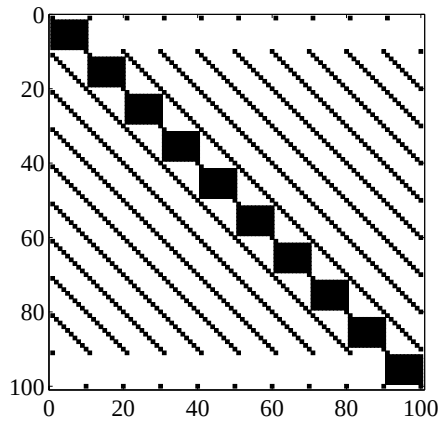
$$\begin{aligned} Q_0(x,y)\frac{\partial^{(n)} f}{\partial x^{(n)}} + Q_1(x,y)\frac{\partial^{(n-1)} f}{\partial x^{(n-1)}} + \ldots + Q_{n-1}(x,y)\frac{\partial f}{\partial x} + Q_n(x,y)f \\ P_0(x,y)\frac{\partial^{(n)} f}{\partial y^{(n)}} + P_1(x,y)\frac{\partial^{(n-1)} f}{\partial y^{(n-1)}} + \ldots + P_{n-1}(x,y)\frac{\partial f}{\partial y} + P_n(x,y)f = g(x,y) \end{aligned} \tag{3.56}$$



a) The operator matrix without boundary conditions applied



b) The operator matrix with Dirichlet boundary conditions on all boundaries



c) The operator matrix with Neumann boundary conditions on all boundaries

Figure 3.10: The location of nonzero elements in the product tensor matrix.

with

$$\begin{cases} K_0(A,y)\dfrac{\partial^{(n)}f}{\partial y^{(n)}} + K_1(A,y)\dfrac{\partial^{(n-1)}f}{\partial y^{(n-1)}} + \ldots + K_{n-1}(A,y)\dfrac{\partial f}{\partial y} + K_n(A,y)f = a(x,y) \\ J_0(B,y)\dfrac{\partial^{(n)}f}{\partial x^{(n)}} + J_1(B,y)\dfrac{\partial^{(n-1)}f}{\partial x^{(n-1)}} + \ldots + J_{n-1}(B,y)\dfrac{\partial f}{\partial x} + J_n(B,y)f = b(x,y) \\ H_0(x,C)\dfrac{\partial^{(n)}f}{\partial y^{(n)}} + H_1(x,C)\dfrac{\partial^{(n-1)}f}{\partial y^{(n-1)}} + \ldots + H_{n-1}(x,C)\dfrac{\partial f}{\partial y} + H_n(x,C)f = c(x,y) \\ G_0(x,D)\dfrac{\partial^{(n)}f}{\partial x^{(n)}} + G_1(x,D)\dfrac{\partial^{(n-1)}f}{\partial x^{(n-1)}} + \ldots + G_{n-1}(x,D)\dfrac{\partial f}{\partial x} + G_n(x,D)f = d(x,y) \\ \quad \vdots \end{cases} \tag{3.57}$$

where the set of boundary conditions is adjusted appropriately to define a well-posed problem. Although Eq. (3.57) is presented in an elaborate general form, it aids to formally and completely detail the solution to such problems. First we write Eq. (3.56) in the operator form

$$\begin{aligned} \Bigg[Q_0(x,y)\frac{\partial^{(n)}}{\partial x^{(n)}} + Q_1(x,y)\frac{\partial^{(n-1)}}{\partial x^{(n-1)}} + \ldots + Q_{n-1}(x,y)\frac{\partial}{\partial x} + Q_n(x,y) \\ P_0(x,y)\frac{\partial^{(n)}}{\partial y^{(n)}} + P_1(x,y)\frac{\partial^{(n-1)}}{\partial y^{(n-1)}} + \ldots + P_{n-1}(x,y)\frac{\partial}{\partial y} + P_n(x,y) \Bigg] f = g(x,y) \end{aligned} \tag{3.58}$$

with

$$\begin{cases} \left[K_0(A,y)\dfrac{\partial^{(n)}}{\partial y^{(n)}} + K_1(A,y)\dfrac{\partial^{(n-1)}}{\partial y^{(n-1)}} + \ldots + K_{n-1}(A,y)\dfrac{\partial}{\partial y} + K_n(A,y)\right] f = a(x,y) \\ \left[J_0(B,y)\dfrac{\partial^{(n)}}{\partial x^{(n)}} + J_1(B,y)\dfrac{\partial^{(n-1)}}{\partial x^{(n-1)}} + \ldots + J_{n-1}(B,y)\dfrac{\partial}{\partial x} + J_n(B,y)\right] f = b(x,y) \\ \left[H_0(x,C)\dfrac{\partial^{(n)}}{\partial y^{(n)}} + H_1(x,C)\dfrac{\partial^{(n-1)}}{\partial y^{(n-1)}} + \ldots + H_{n-1}(x,C)\dfrac{\partial}{\partial y} + H_n(x,C)\right] f = c(x,y) \\ \left[G_0(x,D)\dfrac{\partial^{(n)}}{\partial x^{(n)}} + G_1(x,D)\dfrac{\partial^{(n-1)}}{\partial x^{(n-1)}} + \ldots + G_{n-1}(x,D)\dfrac{\partial}{\partial x} + G_n(x,D)\right] f = d(x,y) \\ \quad \vdots \end{cases} \tag{3.59}$$

For a multidimensional partial differential equation, each independent variable must be mapped over the interval $[-1,1]$. In the two-dimensional case illustrated here we define two general mapping equations as

$$\begin{cases} x = \dfrac{B}{2}(\xi+1) - \dfrac{A}{2}(\xi-1) \quad \longleftrightarrow \quad \xi = \dfrac{2x-(B+A)}{B-A} \\ y = \dfrac{D}{2}(\eta+1) - \dfrac{C}{2}(\eta-1) \quad \longleftrightarrow \quad \eta = \dfrac{2y-(D+C)}{D-C} \end{cases} \tag{3.60}$$

hence

$$\begin{cases} \dfrac{\partial}{\partial x} = \dfrac{2}{B-A}\dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial y} = \dfrac{2}{D-C}\dfrac{\partial}{\partial \eta} \end{cases} \tag{3.61}$$

Next, we express the operator as a spectral matrix. However, the spectral form here is significantly different than that for an ODE to smoothly interact with the tensor product grid. Here we have x, y, $f(x,y)$, and $g(x,y)$ written as

$$x_i = \begin{bmatrix} x_N \\ \vdots \\ x_N \\ x_{N-1} \\ \vdots \\ x_{N-1} \\ \vdots \\ x_1 \\ \vdots \\ x_1 \end{bmatrix}, \; y_j = \begin{bmatrix} y_N \\ y_{N-1} \\ \vdots \\ y_1 \\ y_N \\ y_{N-1} \\ \vdots \\ y_1 \\ y_N \\ \vdots \end{bmatrix}, \; f_i = \begin{bmatrix} f(x_N, y_N) \\ f(x_N, y_{N-1}) \\ \vdots \\ f(x_N, y_1) \\ f(x_{N-1}, y_N) \\ f(x_{N-1}, y_{N-1}) \\ \vdots \\ f(x_{N-1}, y_1) \\ f(x_{N-2}, y_N) \\ \vdots \end{bmatrix}, \text{ and } g_i = \begin{bmatrix} g(x_N, y_N) \\ g(x_N, y_{N-1}) \\ \vdots \\ g(x_N, y_1) \\ g(x_{N-1}, y_N) \\ g(x_{N-1}, y_{N-1}) \\ \vdots \\ g(x_{N-1}, y_1) \\ g(x_{N-2}, y_N) \\ \vdots \end{bmatrix}, \quad (3.62)$$

respectively, where all vectors are length N^2. We must keep in mind that the function `chebdif` returns collocation points from right to left. This is doubly significant with two independent variables - each being written from their right bound to the left. We see that the independent variables are actually written as the column vector $\{x_N, \ldots, x_N, \ldots, x_1 \ldots, x_1\}$ and $\{y_N, y_{N-1}, \ldots, y_1\}$ repeated N times. Again, this is a requirement of the tensor product grid. MATLAB's built-in syntax can achieve these definitions with the commands

```
[xx,y] = meshgrid(x,y);
xx = xx(:); yy = yy(:);
```

Even though the final solution will be expressed as the matrix $f_{i,j}$, the form of the discretized equation must still be $A_{i,j} f_i = B_i$. For this reason the solution is calculated as a vector and then converted into a matrix. In MATLAB this is done with the command `reshape(f,N,N)`. The vector B_i is constructed from the discretized forcing function, $g(x, y)$. The correct vector form of this term is automatically obtained if the function is defined in terms of x_i and y_i.

Any constant coefficients can be multiplied with impunity. Variable coefficients must be written as diagonal matrices. Again, these coefficients will automatically match the tensor product form if they are computed in terms of x_i and y_i. In general, variable coefficients will have the form

$$(Q_j)_{ii} = \begin{bmatrix} q_j(x_N, y_N) & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & q_j(x_0, y_0) & \ddots & & & \vdots \\ \vdots & & & \ddots & q_j(x_n, y_n) & & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & q_j(x_1, y_1) \end{bmatrix} \tag{3.63}$$

where $q_j(x_i, y_i)$ is the spectral elements of any general function and x_i and y_j follow the convention defined above. The complete coefficient matrix will be $N^2 \times N^2$. Equation (3.56) is defined spectrally as

$$\begin{aligned} \big[(Q_0)_{ii}(\bar{D}_N^x)^n + (Q_1)_{ii}(\bar{D}_N^x)^{n-1} + \ldots + (Q_{n-1})_{ii}\bar{D}_N^x + Q_n(x,y)I_N \\ (P_0)_{ii}(\bar{D}_N^y)^n + (P_1)_{ii}(\bar{D}_N^y)^{n-1} + \ldots + (P_{n-1})_{ii}\bar{D}_N^y + P_n(x,y)\big]\, f_i = g_i \end{aligned} \tag{3.64}$$

Now it is easily seen that this spectral equation can be written as

$$A_{ij} f_i = B_i \tag{3.65}$$

where A_{ij} is the two-dimensional operator matrix and B_i is the forcing function in vector form defined over the tensor product grid.

Even though the equation is discretized to a tensor product form, the boundary conditions must be imposed at the matrix elements corresponding to the original interval boundaries. In practice this means that boundary conditions are not applied by rewriting the first and last rows of the operator matrix as in the case of an ODE, but rather all rows containing entries corresponding to the boundaries before the operators are written in tensor product form. This can be coded in two steps: 1) find the elements corresponding to the boundaries in both x and y and 2) rewrite those rows in the operator matrix with the appropriate condition. The first step can be accomplished with

```
Axbc = find(xx==Ax); Bxbc = find(xx==Bx);  % Find and store the locations
                                                 % of the x boundaries in
                                                 % order to find the points
                                                 % in the operator matrix
Aybc = find(yy==Ay); Bybc = find(yy==By);  % Find and store the locations
                                                 % of the y boundaries in
                                                 % order to find the points
                                                 % in the operator matrix
```

The MATLAB logical command `find()` allows us to identify the boundaries without looping over all elements where `Ax,Bx,Ay,By` define the boundaries. It enables us to compute the boundaries on the generated grid, xx and yy, by finding and storing the indices of the locations on the interval bounds. These can be later used to directly impose boundary conditions at those locations.

With regards to the types of boundary conditions, we must consider:

1. For a boundary value problem, the Dirichlet boundary conditions are applied by rewriting the appropriate rows of the operator matrix with the corresponding rows

of the N^2 identity matrix. For instance

$$\begin{cases} f(A,y) = a(x,y) \\ f(B,y) = b(x,y) \\ f(x,C) = c(x,y) \\ f(x,D) = d(x,y) \end{cases} \longrightarrow \tag{3.66}$$

```
% Define the identity matrix for boundary conditions
II=eye(N^2);
Amat(Axbc,:) = II(Axbc,:);
Amat(Bxbc,:) = II(Bxbc,:);
Amat(Aybc,:) = II(Aybc,:);
Amat(Bybc,:) = II(Bybc,:);
```

2. For any Neumann (derivative) boundary conditions, the operator matrix is modified by

$$\begin{cases} f_x(A,y) = a(x,y) \\ f_x(B,y) = b(x,y) \\ f_y(x,C) = c(x,y) \\ f_y(x,D) = d(x,y) \end{cases} \longrightarrow \tag{3.67}$$

```
% Define the differentiation matrix for boundary conditions
DDy=kron(I,d_dy);DDx=kron(d_dx,I);
Amat(Axbc,:) = DDx(Axbc,:);
Amat(Bxbc,:) = DDx(Bxbc,:);
Amat(Aybc,:) = DDy(Aybc,:);
```

```
Amat(Bybc,:) = DDy(Bybc,:);
```

3. For initial value problems we must overwrite the rows in the operator matrix containing elements corresponding to both the first and last rows of the original domain (i.e. the same rows of the operator matrix as for boundary value problems) with the rows of the identity and derivative matrices corresponding to the left bound. This condition warrants

$$\begin{cases} f(A,y) = a(x,y) \\ f_x(A,y) = b(x,y) \\ f(x,C) = c(x,y) \\ f_y(x,C) = d(x,y) \end{cases} \longrightarrow \tag{3.68}$$

```
% Define the identity matrix for boundary conditions
II=eye(N^2);
% Define the differentiation matrix for boundary conditions
DDy=kron(I,d_dy);DDx=kron(d_dx,I);
Amat(Axbc,:) = II(Axbc,:);
Amat(Bxbc,:) = DDx(Axbc,:);
Amat(Aybc,:) = II(Aybc,:);
Amat(Bybc,:) = DDy(Aybc,:);
```

Higher order derivatives are defined accordingly.

3.6.1 Example: A Parabolic PDE with Variable Coefficients

This section is designed to illustrate the solution of a parabolic partial differential equation, similar to the heat equation, with variable coefficients as well as the inclusion of the undifferentiated function in the original equation. Also, the equation is characterized by

derivative boundary conditions to illustrate their proper inclusion into the operator matrix. We consider the equation

$$f_y = \cos(y(1-x))f_{xx} - xf \tag{3.69}$$

with boundary conditions

$$\begin{cases} f(x,0) = -2\cos(\pi x) \\ f_x(0,y) = 0 \\ f_x(1,y) = -2 \end{cases} \tag{3.70}$$

To begin we must specify the number of collocation points. The number can be different in each direction and it might be advantageous to do so if one direction requires much less resolution. Conversely, it will require considerable care when dealing with the ensuing matrix operations, boundary conditions, and eigenvalue calculations. Furthermore, we would have to carry more variables throughout the program, thus increasing hardware requirements. For these reasons, only equal numbers of collocation points in both directions will be considered. We define the number of points, the interval and, the derivatives with

```
N = 20;         % Define the number of points in the x and y directions
Ax=-1;Bx=1;      % Define the left and right bounds in x, respectively
Ay=0;By=1;      % Define the left and right bounds in y, respectively
[xi,D] = chebdif(N,1);       % Define the x collocation points
   eta = xi;                 % Define the y collocation points
x=Bx/2*(xi+1)-Ax/2*(xi-1);   % Convert back to the original domain
y=By/2*(xi+1)-Ay/2*(xi-1);   % Convert back to the original domain
d_dx=2/(Bx-Ax)*D(:,:,1);     % Convert the x Spectral derivatives
d_dy=2/(By-Ay)*D(:,:,1);     % Convert the y Spectral derivatives
```

Since the number of collocation points is the same in both directions we could simply define one derivative and one collocation grid but for illustrative purposes we will clearly define each for both x and y directions.

Thus far the coding has been nearly the same as for the solution to an ODE. The command `[xx,yy]=meshgrid[x,y]` takes the vector x and builds an $N \times N$ matrix, xx where the columns are built by the vector x. Likewise, the rows of yy are built from the vector y. Simple as it seems, this command builds the x and y grid in which the solution will be interpolated over. The code looks like

```
[xx,yy] = meshgrid(x,y);     % Mesh the grid
xx = xx(:); yy = yy(:);      % Convert the meshed grid to vectors to match
                                 % tensor product form of the governing eq.
                                 % and map the boundary conditions
```

Next we define the operator matrix via Kronecker products.

```
II=eye(N^2);                  % Define an identity matrix the size of Amat
% Define the spectral operator matrix
Amat = kron(I,d_dy) -diag(cos((1-xx).*yy)) * kron(d2_dx2,I)+diag(xx)*II;
```

: refer to Eq. (3.69)

Note the inclusion of the **diag**() command and the use of the column vectors, xx and yy, to properly define all variable coefficients.

Per the opening discussion on boundary conditions, we use the MATLAB logical command **find**() to identify the boundaries automatically. We code

```
Axbc = find(xx==Ax); Bxbc = find(xx==Bx);  % Find and store the locations
                                                % of the x boundaries in
                                                % order to find the points
                                                % in the operator matrix
Aybc = find(yy==Ay); Bybc = find(yy==By);  % Find and store the locations
                                                % of the y boundaries in
```

```
                                                % order to find the points
                                                % in the operator matrix
```

The conditions are imposed with

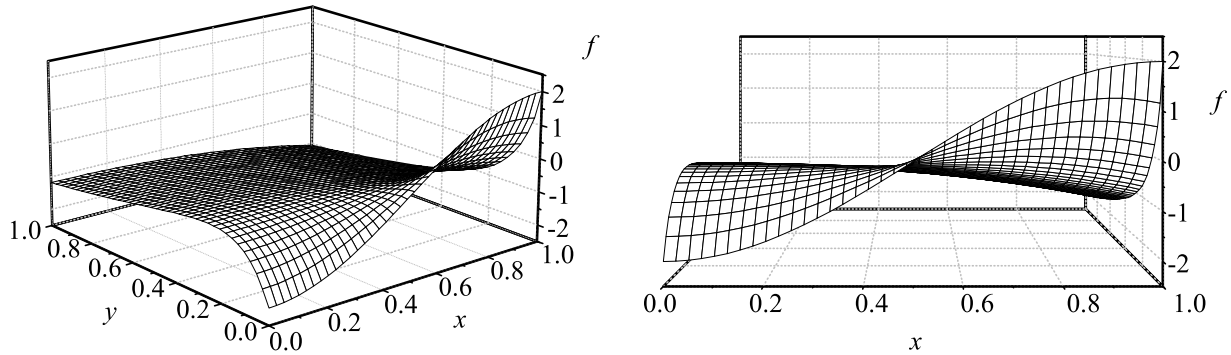
```
% Define the differentiation matrix for boundary conditions
DDy=kron(I,d_dy);DDx=kron(d_dx,I);
Amat(Axbc,:) = DDx(Axbc,:);
Amat(Bxbc,:) = DDx(Bxbc,:);
Amat(Aybc,:) = II(Aybc,:);
% Amat(Bybc,:) = II(Bybc,:);
```

: refer to Eq. (3.70)

Since this equation is first order in y we can only include one boundary condition. Here we define the left interval. Defining both intervals leads to an overdetermined system typically marred by numeric instability or incorrect values at the boundaries.

Next we can define the right hand side of the equation containing any forcing functions and boundary conditions. This equation does not have a forcing function so the right hand side consist zeros except for those elements defined by the boundary conditions. For this equation we have

```
Bvec = zeros(N^2,1);       % Define the forcing function as the B vector
% Define u(x,y) at x=Ax
Bvec(Axbc) = 0;
% Define u(x,y) at x=Bx
Bvec(Bxbc) = -2;
% Define u(x,y) at y=Ay
Bvec(Aybc) = -2*cos(pi*xx(Aybc));%sin(pi*xx(Aybc));
% Define u(x,y) at y=By
```



a) Three dimensional view

b) Profile view to show the boundary conditions

Figure 3.11: The solution to the given parabolic PDE under the boundary conditions given above for $N = 20$.

```
% Bvec(Bybc) = 0;
```

: refer to Eq. (3.70)

What is left is to solve the equation and plot the solution over a refined grid. The solution is found as usual by `f=Amat\Bvec`. This produces the solution as a vector. We write it in matrix form by the command `ff=reshape(f,N,N)`. Now the solution can be interpolated over a finer grid space in each direction independently. Each row is considered to interpolate in the x direction then each column is in turn interpolated to refine the y direction. The code to do this is

```
for i=1:N
    P(i,:)=chebint(ff(i,:),linspace(-1,1,xgrid));
end
for j=1:xgrid
    PP(:,j)=chebint(P(:,j),linspace(-1,1,ygrid));
end
```

Finally, the solution can be plotted with the command `mesh(xxx,yyy,PP)`. The complete code is given in Alg. B.10.1 on page 308. Figure 3.11 shows that the boundary conditions

are satisfied on all edges. This plot was discretized over $N = 20$ collocation points and interpolated over a 40×40 grid.

3.6.2 Example: The 2D Poisson Equation

We will continue with an example from Trefethen [95]. In this example we will consider how to handle forcing functions with the example of the two-dimensional Poisson equation. This equation is taken exactly as is from Trefethen, but solved using our code as developed above. We are considering the equation

$$f_{xx} + f_{yy} = 10 \sin\left(8x(y-1)\right), \qquad -1 \leq x, y \leq 1, \quad f = 0 \ \text{ at all boundaries} \tag{3.71}$$

The coding of this solution is the same as shown in the previous example with the following differences. The operator matrix now appears as

```
Amat = kron(d2_dx2,I) + kron(I,d2_dy2);
```

: refer to Eq. (3.71)

and the B vector becomes

```
Bvec=10*sin(8*xx.*(yy-1))
```

: refer to Eq. (3.71)

From here, we let all of the boundary conditions equal zero with

```
Amat(Axbc,:) = II(Axbc,:); Bvec(Axbc) = 0;
Amat(Bxbc,:) = II(Bxbc,:); Bvec(Bxbc) = 0;
Amat(Aybc,:) = II(Aybc,:); Bvec(Bybc) = 0;
```

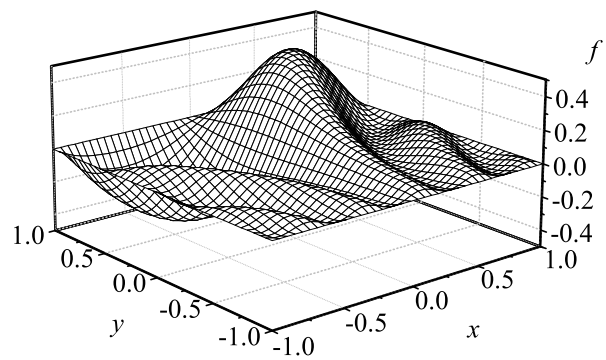


Figure 3.12: The solution to the time-independent Poisson equation with a sinusoidal forcing function under the boundary conditions given above for $N = 20$.

```
Amat(Bybc,:) = II(Bybc,:); Bvec(Aybc) = 0;
```

: refer to Eq. (3.71)

The corresponding solution is plotted in Fig. 3.12. The complete code is included as Alg. B.11.1 on page 312.

3.6.3 Example: A System of PDEs

Systems of PDEs are handled similarly to systems of ODEs while keeping in mind the properties of the tensor product grid. For the system to be defined in a single operator matrix, each equation will be allocated N^2 rows and each dependent variable will be allocated N^2 columns. This requirement is greater for PDEs than the $N \times N$ block matrices required for ODEs since the Kronecker products result in an N^2 operator matrix *for each dependent variable*. One can therefore expect a significant increase in the required computational power when handling systems of equations. To demonstrate the method of solution for systems of

PDEs, we consider the coupled set of equations

$$\begin{cases} f_{xx} + x f_{yy} - g_x = \sin(\pi x) \\ g_{xx} + g_{yy} + f_x = \cos(\pi x) \end{cases} \quad \text{with} \quad \begin{cases} f(0,y) = 0; & f(1,y) = 0 \\ f(x,0) = \sin(\pi x); & f(x,1) = 0 \\ g(0,y) = 0; & g(1,y) = 0 \\ g_y(x,0) = 0; & g(x,1) = 1 \end{cases} \tag{3.72}$$

Next, the equations are written, as usual, in operator form:

$$\begin{cases} \left(\dfrac{\partial^2}{\partial x^2} + x\dfrac{\partial^2}{\partial y^2}\right) f(x,y) & + & (-1)\left(\dfrac{\partial}{\partial x}\right) g(x,y) & = & \sin(\pi x) \\ (1)\left(\dfrac{\partial}{\partial x}\right) f(x,y) & + & \left(\dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}\right) g(x,y) & = & \cos(\pi x) \end{cases} \tag{3.73}$$

with boundary conditions

$$\begin{cases} (1)f(0,y) + (0)f(0,y) + (0)g(0,y) + (0)g(0,y) = 0 \\ (0)f(1,y) + (1)f(1,y) + (0)g(1,y) + (0)g(1,y) = 0 \\ (0)f(0,y) + (0)f(0,y) + (1)g(0,y) + (0)g(0,y) = 0 \\ (0)f(1,y) + (0)f(1,y) + (0)g(0,y) + (1)g(1,y) = 0 \end{cases} \tag{3.74}$$

and

$$\begin{cases} (1)f(x,0) + (0)f(x,0) + (0)g(x,0) + (0)g(x,0) = \sin(\pi x) \\ (0)f(x,1) + (1)f(x,1) + (0)g(x,1) + (0)g(x,1) = 0 \\ (0)f_y(x,0) + (0)f_y(x,0) + (1)g_y(x,0) + (0)g_y(x,0) = 0 \\ (0)f(x,1) + (0)f(x,1) + (0)g(x,1) + (1)g(x,1) = 1 \end{cases} \tag{3.75}$$

From this step we can naturally convert the system to its equivalent matrix form,

$$\begin{bmatrix} \left(\dfrac{\partial^2}{\partial x^2} + x\dfrac{\partial^2}{\partial y^2}\right) & (-1)\left(\dfrac{\partial}{\partial x}\right) \\ (1)\left(\dfrac{\partial}{\partial x}\right) & \left(\dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}\right) \end{bmatrix} \begin{bmatrix} f(x,y) \\ g(x,y) \end{bmatrix} = \begin{bmatrix} \sin(\pi x) \\ \cos(\pi x) \end{bmatrix} \tag{3.76}$$

with boundary conditions expressed as

$$\begin{bmatrix} (1) & (0) & (0) & (0) \\ (0) & (1) & (0) & (0) \\ (0) & (0) & (1) & (0) \\ (0) & (0) & (0) & (1) \end{bmatrix} \begin{bmatrix} f(0,y) \\ f(1,y) \\ g(0,y) \\ g(1,y) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.77}$$

and

$$\begin{bmatrix} (1) & (0) & (0) & (0) \\ (0) & (1) & (0) & (0) \\ (0)\left(\dfrac{\partial}{\partial y}\right) & (0)\left(\dfrac{\partial}{\partial y}\right) & (1)\left(\dfrac{\partial}{\partial y}\right) & (0)\left(\dfrac{\partial}{\partial y}\right) \\ (0) & (0) & (0) & (1) \end{bmatrix} \begin{bmatrix} f(x,0) \\ f(x,1) \\ g(x,0) \\ g(x,1) \end{bmatrix} = \begin{bmatrix} \sin(\pi x) \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.78}$$

It is necessary to map both spatial domains over the interval $[-1, 1]$. We apply Eq. (3.23) to both independent variables to recover

$$\begin{aligned} x = \frac{1}{2}(\xi - 1) \quad &\longleftrightarrow \quad \xi = 2x - 1 \quad \text{with} \quad \frac{\partial}{\partial x} = 2\frac{\partial}{\partial \xi} \\ y = \frac{1}{2}(\eta - 1) \quad &\longleftrightarrow \quad \eta = 2y - 1 \quad \text{with} \quad \frac{\partial}{\partial y} = 2\frac{\partial}{\partial \eta} \end{aligned} \tag{3.79}$$

At this juncture we can represent the operator matrix in spectral notation,

$$\begin{bmatrix} (\bar{D}_N^x)^2 + x_{ii}(\bar{D}_N^y)^2 & -\bar{D}_N^x \\ \bar{D}_N^x & (\bar{D}_N^x)^2 + (\bar{D}_N^y)^2 \end{bmatrix} \begin{bmatrix} f(x_i, y_i) \\ g(x_i, y_i) \end{bmatrix} = \begin{bmatrix} \sin(\pi x) \\ \cos(\pi x) \end{bmatrix} \tag{3.80}$$

It is important to keep in mind that the first column refers to operations on the function $f(x, y)$ and the second column operates on $g(x, y)$. At the outset, the first row constructs the first equation and the second row yields the second. For larger systems we will add columns and rows accordingly. The general scheme will look like

$$A_{ij} = \begin{array}{c} \\ \mathrm{Eq}_1 \\ \vdots \\ \mathrm{Eq}_m \end{array} \begin{array}{c} \\ \rightarrow \\ \\ \rightarrow \end{array} \begin{array}{c} \begin{array}{ccc} f_1(x,y) & \cdots & f_m(x,y) \\ \downarrow & & \downarrow \end{array} \\ \left[\begin{array}{c|c|c} A_{1,f_1} & \cdots & A_{1,f_m} \\ \hline \vdots & \ddots & \vdots \\ \hline A_{m,f_1} & \cdots & A_{m,f_m} \end{array}\right] \end{array} \tag{3.81}$$

where m is the number of equations and unknowns and $A_{m,f}$ are block operator matrices for each variable and equation. Even though PDEs are expressed over a tensor product grid, the operator matrix has the same for as that of ODEs, the difference being in the definitions of the derivatives and coefficients. The matrix is constructed for this set of equations by

```
j=1:1:N^2;
% Define the operator on f from Eq.1
Amat(j,j) = kron(d2_dx2,I)+ diag(xx)*kron(I,d2_dy2);
% Define the operator on g from Eq.1
Amat(j,N^2+j) = -kron(d_dx,I);
% Define the operator on f from Eq.2
Amat(N^2+j,j) = kron(d_dx,I);
% Define the operator on g from Eq.2
```

```
Amat(N^2+j,N^2+j) = kron(d2_dx2,I)+ kron(I,d2_dy2);
```

: refer to Eq. (3.80)

Boundary conditions require significant care to implement. Recall in the example for systems of ODEs that each boundary condition operator for a specific function is applied to the rows of a corresponding block matrix while the same rows of the horizontally adjacent block matrix are zeroed. In the solution of a single PDE, the elements along the boundary are identified within the tensor product matrix and boundary condition operators are applied across their rows. Therefore boundary condition operators are applied to a system of PDEs by combining those two ideas. First the boundary elements are identified. Next the boundary condition operators are applied to the rows of the correct block matrix while the horizontally adjacent block matrix is zeroed. In the example given above, the order of the differential equation is 2 for both x and y; thus we see 8 total boundary conditions (4 applied to each of $f(x, y)$ and $g(x, y)$). Thus, we apply 4 boundary conditions (the 4 conditions on $f(x, y)$) to the top two block matrices and 4 (the 4 conditions on $g(x, y)$) to the bottom two. The conditions could be applied differently and at the discretion of the user; however, this convention is proven effective, as illustrated in the code below

```
% f(0,y) and f(1,y)
Amat(Axbc,j) = II(Axbc,:);        % Define the IV operator on f from BC on f
Amat(Axbc,N^2+j)=0*II(Axbc,:);    % Define the IV operator on g from BC on f
Bvec(Axbc) = 0;                                          % Define the IV on f
Amat(Bxbc,j) = II(Bxbc,:);        % Define the BC operator on f from BC on f
Amat(Bxbc,N^2+j)=0*II(Bxbc,:);    % Define the BC operator on g from BC on f
Bvec(Bxbc) = 0;                                          % Define the BC on f

% f(x,0) and f(x,1)
Amat(Aybc,j) = II(Aybc,:);        % Define the IV operator on f from BC on f
Amat(Aybc,N^2+j)=0*II(Aybc,:);    % Define the IV operator on g from BC on f
```

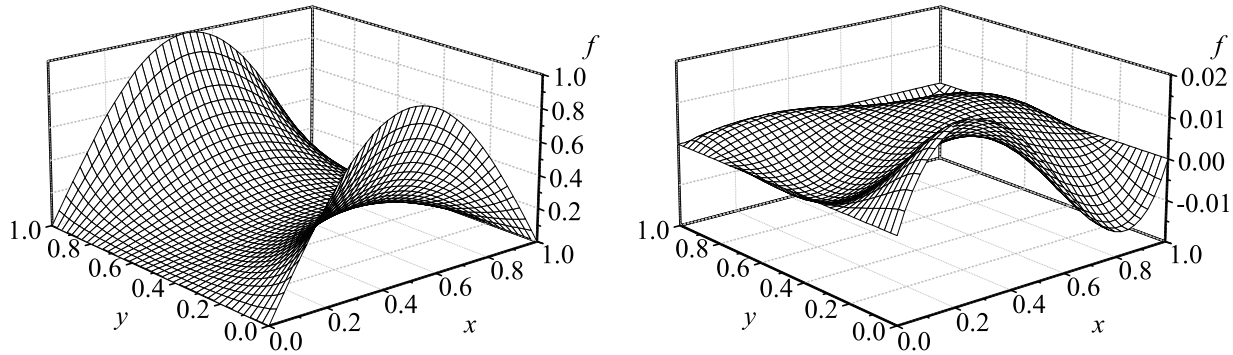
```
Bvec(Aybc) = sin(pi*xx(Aybc));                          % Define the IV on f
Amat(Bybc,j) = II(Bybc,:);        % Define the BC operator on f from BC on f
Amat(Bybc,N^2+j)=0*II(Bybc,:);    % Define the BC operator on g from BC on f
Bvec(Bybc) = sin(pi*xx(Bybc));                          % Define the BC on f

% g(0,y) and g(1,y)
Amat(N^2+Axbc,j)=0*II(Axbc,:);    % Define the IV operator on f from BC on g
Amat(N^2+Axbc,N^2+j)=II(Axbc,:); % Define the IV operator on g from BC on g
Bvec(N^2+Axbc)=0;                                       % Define the IV on g
Amat(N^2+Bxbc,j)=0*II(Bxbc,:);    % Define the BC operator on f from BC on g
Amat(N^2+Bxbc,N^2+j)=II(Bxbc,:); % Define the BC operator on g from BC on g
Bvec(N^2+Bxbc)=0;                                       % Define the BC on g

% g(x,0) and g(x,1)
Amat(N^2+Aybc,j)=0*DDy(Aybc,:);  % Define the IV operator on f from BC on g
Amat(N^2+Aybc,N^2+j)=DDy(Aybc,:);% Define the IV operator on g from BC on g
Bvec(N^2+Aybc)=0;                                       % Define the IV on g
Amat(N^2+Bybc,j)=0*II(Bybc,:);    % Define the BC operator on f from BC on g
Amat(N^2+Bybc,N^2+j)=II(Bybc,:); % Define the BC operator on g from BC on g
Bvec(N^2+Bybc)=0;                                       % Define the BC on g
```

: refer to Eq. (3.78)

The complete code is found in Alg. B.12.1 on page 315. The solution is plotted for $N = 20$ in Fig. 3.13. The solutions possess an error of 6.5409E-012 and 2.8512E-012 for $f(x, y)$ and $g(x, y)$, respectively.

3.7 Eigenvalue Problems with PDEs

Much of the discussion regarding eigenvalue problems has been covered in the ODE section. As per norm, eigenvalue problems involving PDEs are more difficult mainly due to the



a) $f(x, y)$ **b)** $g(x, y)$

Figure 3.13: The solution to the system of PDEs under the boundary conditions given above for $N = 20$.

product tensor grid and the application of boundary conditions. However, we have already developed all of the tools necessary to properly write and solve these problems spectrally.

3.7.1 Example: The Helmholtz equation

In this section we will consider the eigenvalue problem posed by the Helmholtz equation in two dimensions. It is

$$f_{xx} + f_{yy} + \mu f = 0, \quad 0 \leq x, y \leq L, \quad f = 0 \text{ on all boundaries} \tag{3.82}$$

This equation is solvable via separation of variables and results in the eigenfunction

$$\sin(k_x x)\sin(k_y y) \tag{3.83}$$

where k_x and k_y are integer multiples of π/L and the eigenvalues are given by

$$\mu = \frac{\pi^2}{L^2}(i^2 + j^2); \quad i, j = 0, 1, 2... \tag{3.84}$$

To proceed, we segregate terms as usual to facilitate a spectral representation. We have

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) f(x,y) + \lambda f(x,y) = 0 \tag{3.85}$$

with $\lambda = \mu$. Substituting the spectral derivative operators into this equation allows us to write the Helmholtz equation as a generalized eigenvalue problem (Eq. (3.45)) such that

$$A_{ij} = (\bar{D}_N^x)^2 + (\bar{D}_N^y)^2 \quad \text{and} \quad B_{ij} = I_N \tag{3.86}$$

These matrices are easily constructed with

```
Amat = kron(d2_dx2,I) + kron(I,d2_dy2);
Bmat = II;
```

: refer to Eq. (3.86)

The boundary conditions maybe imposed as before by determining the elements within the product tensor matrix that appear on the boundaries and apply the boundary conditions accordingly. We use

```
Amat(Axbc,:) = II(Axbc,:);
Amat(Bxbc,:) = II(Bxbc,:);
Amat(Aybc,:) = II(Aybc,:);
Amat(Bybc,:) = II(Bybc,:);

Bmat(Axbc,:) = 0*II(Axbc,:);
Bmat(Bxbc,:) = 0*II(Bxbc,:);
Bmat(Aybc,:) = 0*II(Bybc,:);
```

```
Bmat(Bybc,:) = 0*II(Bybc,:);
```

: refer to Eq. (3.82)

Lastly, the eigenvalues are solved numerically. The eigenvalues can be compared over an arbitrary domain by considering the group $\mu L^2/\pi^2$. The example for ODEs is normalized by multiplying by the domain interval only once. The extra L needed here is due to the inclusion of the second spatial dimension.

The complete spectral code is presented in Alg. B.13.1 on page 321. The normalized analytic eigenvalues are compared to the numeric results in Table 3.3. Note that the error does not increase smoothly with successive increases in the eigenvalue. The product tensor grid changes the order in which the eigenvalues are computed; thus, they are not computed from the smallest to the largest as usual. Additionally, the accuracy persists further down the spectrum for smaller values of N than in the previous ODE example. From this perspective, we can expect smaller values of N to provide comparable spectral resolution for biglobal stability calculations than for classic one-dimensional approaches.

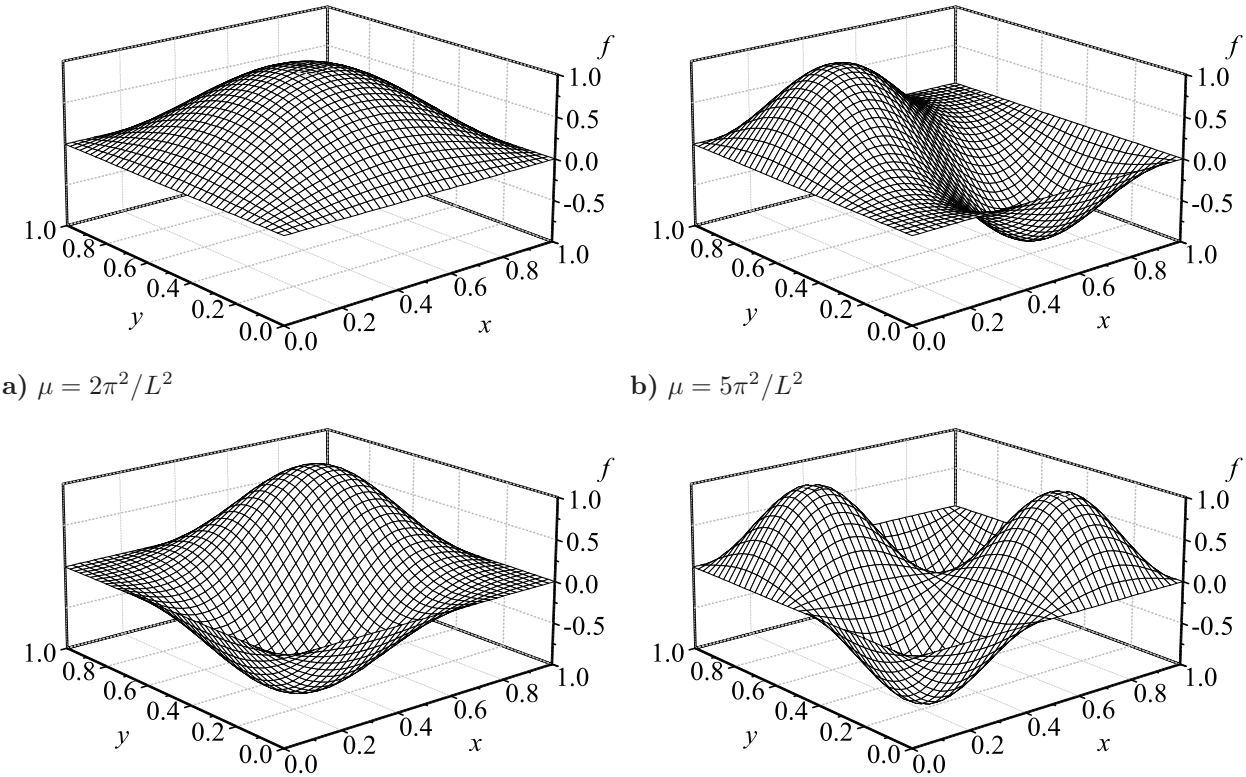
3.8 Closing Remarks on Spectral Methods

Our discussion on spectral methods has only dealt with linear equations. Since the hydrodynamic stability equations are linear, this course of discussion seems appropriate. It is important to recognize however, that spectral methods also work well with nonlinear problems. For these problems we must implement a convergence scheme by choosing an initial guess, v, solving the equation with the nonlinear terms appearing as a forcing function in the B vector, updating v to be the new initializing vector, and iterating until convergence occurs. An example is the equation

$$u_{xx} = e^u \tag{3.87}$$

Table 3.3: The normalized eigenvalues of the Helmholtz equation for $0 \leq x \leq L$ compared to their exact values. $\% \ E = |\lambda_e - \mu|/\lambda_e$ where λ_e is the exact value.

$\lambda_e \frac{L^2}{\pi^2}$	$\mu \frac{L^2}{\pi^2}$							
	$N = 5$	$\% \ E$	$N = 10$	$\% \ E$	$N = 15$	$\% \ E$	$N = 20$	$\% \ E$
2	1.9908	0.4622	2.0000	1.440E-06	2.0000	4.496E-12	2.0000	1.500E-11
5	5.8588	17.1759	5.0000	0.0002	5.0000	8.620E-09	5.0000	5.596E-12
5	5.8588	17.1759	5.0000	0.0002	5.0000	8.618E-09	5.0000	3.393E-12
8	8.9163	11.4533	8.0000	0.0003	8.0000	1.077E-08	8.0000	4.130E-12
10	8.9163	10.8374	10.0024	0.0238	10.0000	8.201E-07	10.0000	7.130E-11
10	9.7268	2.7317	10.0024	0.0238	10.0000	8.201E-07	10.0000	7.031E-11
13	12.7843	1.6592	13.0024	0.0183	13.0000	6.341E-07	13.0000	5.538E-11
13	12.7843	1.6592	13.0024	0.0183	13.0000	6.341E-07	13.0000	5.076E-11
17	15.8418	6.8131	16.9193	0.4748	17.0000	0.0001	17.0000	4.890E-09
17			16.9193	0.4748	17.0000	0.0001	17.0000	4.886E-09
18			18.0048	0.0265	18.0000	0.0000	18.0000	7.723E-11
20			19.9193	0.4036	20.0000	0.0001	20.0000	4.155E-09
20			19.9193	0.4036	20.0000	0.0001	20.0000	4.153E-09
25			24.9217	0.3133	25.0000	0.0001	25.0000	3.354E-09
25			24.9217	0.3133	25.0000	0.0001	25.0000	3.350E-09
26			27.4411	5.5428	26.0001	0.0003	26.0000	7.013E-07
26			27.4411	5.5428	26.0001	0.0003	26.0000	7.013E-07
29			30.4411	4.9694	29.0001	0.0003	29.0000	6.287E-07
29			30.4411	4.9694	29.0001	0.0003	29.0000	6.287E-07
32			31.8386	0.5044	32.0000	0.0001	32.0000	5.194E-09



a) $\mu = 2\pi^2/L^2$ **b)** $\mu = 5\pi^2/L^2$

c) $\mu = 5\pi^2/L^2$ **d)** $\mu = 8\pi^2/L^2$

Figure 3.14: The eigenvectors and their corresponding eigenvalues for the Helmholtz equation for $N = 20$. Figures 3.14b–3.14c show the eigenvectors for the double eigenvalue $\mu = 5\pi^2/L^2$.

An initial guess is defined as u_i and the equation is written spectrally as $(D_N)^2 u_{i+1} = \exp u_i$. This equation is repeatedly solved for u_{i+1} with each iteration considering the updated u vector until some convergence criterion is satisfied.

In certain situations where the problem is sensitive to the number of collocation points, N, and where N needs to be very large, it may be important to iterate on N to ensure convergence. Basic error analysis can be done a priori and an estimate for the collocation number can be determined [99]. Alternatively, convergence can be found iteratively by successively increasing the collocation number in the code. Iterative convergence can be done through brute force by incrementally increasing N until the solution no longer changes. This idea can be much improved by implementing a root finding scheme similar to that used

in shooting methods for boundary value problems. In this case we consider the equation

$$N_{i+1} = \text{round}\left(g(N_i)\frac{N_i - N_{i-1}}{g(N_i) - g(N_{i-1})}\right) \tag{3.88}$$

where $g(N_i) = \max|f(N_i) - f(N_{i-1})|$ and $f(N)$ is the polynomial solution to the governing equation for a given N. This is simply the secant method applied to the equation $g(N)$. It requires two initial values, N_i and N_{i-1} and a convergence criterion that stops the iteration when either a specified tolerance for $g(N_i)$ is met or when $N_{i+1} = N_i$. For many (most) problems this may not be necessary and for problems with extremely large N this might be computationally taxing. However, when all sources of potential error must be minimized this approach can be invaluable.

There are several sources of error to be considered with collocation methods and, correspondingly, several tests to ensure accuracy. The obvious test is to compare the numerical solution to the exact result, if one exists. This may be accomplished by reconstructing the numerical equivalent of the governing equation by differentiating the numeric solution and substituting it back into the governing equation. This test can be neatly applied to any equation or system of equations and is especially helpful for those whose exact solution is not known. It is best applied to the inner nodes for PDEs since arbitrary boundary conditions can cause numeric stability problems along the ends of the interval. Of course, boundary conditions can be verified independently as well. Lastly, solving the matrix $A_{ij}f_i = B_i$ can be a source of numeric error. This is especially true if B is singular or nearly singular. In general, this error increases as N increases and as the number of equations in the system increases because larger matrices tend to accumulate larger round-off error. It is easily quantified by computing $A_{ij}f_i = B_i$ and determining its maximum error. Given these various avenues for verification, most if not all of the methods qualifying error are calculated in the codes included in the appendix.

Those familiar with collocation methods may note that the final solution is plotted over a square grid rather than over the Chebyshev points themselves. This is made possible by the implementation of the barycentric formula in `chebint` which allows robust interpolation of the solution over any collocation grid. If problems with grid spacing do arise, then the solution can be plotted as is without grid refinement. Further interpolation is simply a way to smooth coarse solutions without requiring the extra computational time needed to directly compute the solution over a fine grid from the start.

Lastly, Ch. 4 steps through the mathematics and code development. Ultimately the final codes produced over the course of the chapter are purely academic in that they cannot compete with MATLAB's built-in LAPACK routines in terms of computational time. They serve only to archive the understanding of complete eigensolver routines.

Table 3.4: Applicable error values for all examples.

Example	Back Sub.	Max Local E	$f = A_{ij} \backslash B_i$	Left x B.C.	Right x B.C.	Left y B.C.	Right y B.C.
1st Order ODE	5.77E-15	4.04E-04	8.76E-15	0			
2nd Order ODE	7.77E-14	5.33E-10	8.58E-14	0	6.66E-16		
Sys. of ODEs	7.51E-13		7.02E-13	2.84E-14	0		
	3.43E-13			3.55E-15	0		
Bessel	5.63E-10	0		1.61E-12	0		
Eigenvalues	5.14E-10	1.46E-12		1.45E-12	0		
	8.98E-09	2.12E-12		1.58E-12	0		
	1.36E-05	4.62E-10		1.38E-12	0		
Parabolic PDE	2.76E-11		5.81E-11	9.09E-13	3.30E-12	1.63E-11	
Poisson Eqn	1.53E-13		6.10E-13	0	8.16E-14	2.73E-14	3.39E-14
Sys. of PDEs	6.34E-12		1.74E-11	5.48E-14	2.92E-13	1.90E-13	3.14E-13
	1.72E-12			0	1.04E-13	2.49E-14	1.82E-14
Helmholtz	5.37E-11			0	0	0	0
Eigenvalues	5.96E-11			0	0	0	0
	2.36E-11			0	0	0	0
	6.98E-11			0	0	0	0

Chapter 4

Eigensolvers

This chapter aims to detail the road map and construction of efficient eigensolvers. Similar to the discussion of spectral collocation methods, the following review of eigensolvers would not fit in a traditional journal publication. It is, however, important to understand the linear algebra and numerical application of eigensolvers in fulfilling the requirements of a complete study. In an attempt to remain as concise as possible, many elementary points will be omitted or referenced elsewhere while still illuminating the important details. In brief, Fig. 4.1 illuminates the necessary steps in completing an efficient and accurate eigensolver.

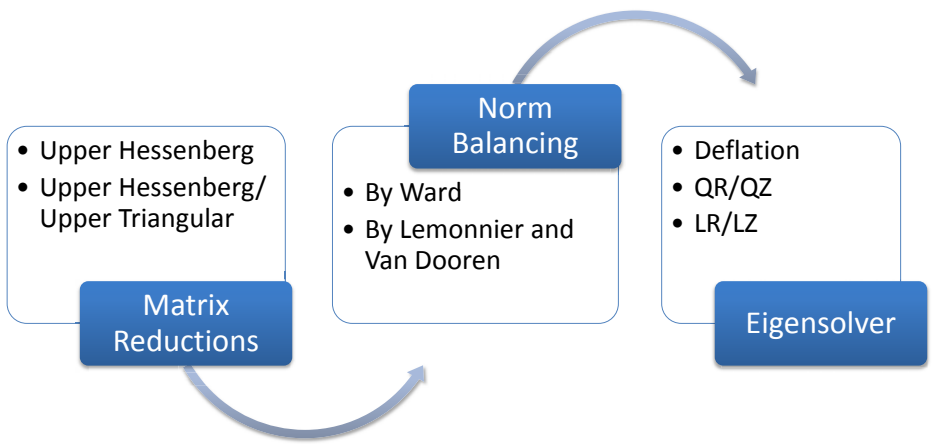


Figure 4.1: Eigensolver Flowchart

4.1 Calculating Eigenvalues

The generalized eigenvalue problem has been defined as

$$A_{ij} f_i = \lambda B_{ij} f_i \tag{4.1}$$

From linear algebra we know that the eigenvalues of a *single matrix* can be found by solving the equation

$$\det(A_{ij} - \lambda I_N) = 0 \tag{4.2}$$

where A_{ij} is any matrix, λ is the eigenvalue and I_N is the identity matrix of size N. From this equation, the *characteristic polynomial* can be computed by expanding the determinant. The N eigenvalues can be explicitly determined by computing the zeros of the Nth order characteristic polynomial. Since the solution of a single matrix eigenvalue problem is known, we should note that the generalized form can be expressed as a single matrix eigenvalue

problem if we multiply through by B_{ij}^{-1}. This action would reformulate the problem as

$$A_{ij}B_{ij}^{-1} - \lambda B_{ij}B_{ij}^{-1} = C_{ij} - \lambda I_N = 0; \quad C_{ij} = A_{ij}B_{ij}^{-1} \tag{4.3}$$

Then the eigenvalues of C_{ij} are the eigenvalues of the original problem. Clearly, B_{ij} cannot be singular for this to work. It is not obvious but from a numerical standpoint, this approach will incur large errors due to the computational accuracy of inverting and solving rather than solving directly [109]. Moreover, inverting large matrices is a computationally expensive task which can be circumvented by constructing an eigensolver that handles the generalized eigenvalue problem directly.

Special cases exist where calculating eigenvalues are extremely easy. Those most important to the problems discussed here include the situation where A_{ij} is upper triangular. In this case, the eigenvalues are simply the diagonal elements. It may hence be seen that, applying similarity transformation matrices* to the matrix A_{ij} in an attempt to convert it to upper triangular form may be more efficient than determining the roots of the characteristic polynomial of arbitrary order N. This notion can be extended further to the generalized eigenvalue problem. If both A_{ij} and B_{ij} are upper triangular, then the generalized eigenvalues are simply a_{ii}/b_{ii}, where a_{ii} and b_{ii} are the diagonal elements of A_{ij} and B_{ij}, respectively. This relationship is applicable to singular and nonsingular matrices alike.

4.2 Matrix Preconditioning

In some cases, numeric eigenvalue calculations are sensitive to rounding errors. These errors are proportional to the sum of the squares of the matrix elements (i.e. the Frobenius norm,

*Two $N \times N$ matrices, A and A^*, are similar if $A^* = Q^{-1}AQ$ where Q is a change-of-basis matrix. A^* can be constructed to have a more useful matrix form (upper Hessenberg, upper triangular, tri-diagonal, etc.) without changing the eigenvalues.

$\|A_{ij}\|_F$) [110].

$$\|A_{ij}\|_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}|a_{ij}|^2} \tag{4.4}$$

Eigenvalues are typically sensitive when their modulus is several orders of magnitude smaller than the norm of the matrix [111]. Therefore, rounding errors can be mitigated if the overall matrix norm is reduced. Matrix balancing uses similarity transformations that cause corresponding rows and columns to have comparable norms, thus reducing the overall Frobenius norm without changing the eigenvalues. This procedure has N^2 operations for each iteration: N operations for rows and N operations for columns. At first it seems time consuming to run balancing, but in comparison the time to execute a matrix balance is small compared to the time to solve for the eigenvalues. Typically appropriate similarity transformations are found in two to three iterations. It is therefore suggested that a balancing procedure always be run.

From experience, it seems that numeric accuracy can be improved by forcing all columns and rows containing only zero elements to the left and up respectively. Further, row and column swapping can be implemented to ensure as many nonzero diagonal elements as possible. This becomes important when dealing with large and/or sparse matrices. Dense matrices usually do not have columns or rows of this type so it is not of much concern. If these ideas are used as a preconditioner, matrix reductions will keep all (or most) of the zero diagonal elements together. Finding zeros on the diagonal leads to division by zero in the *bulge chasing* technique, commonly used to determine the eigenvalues. Furthermore, a matrix can be decomposed if there are blocks of nonzero elements arranged around the diagonal and surrounded by zero elements. If this is the case, then the eigenvalues of the submatrix of nonzero elements become eigenvalues of the total matrix and can be treated as such. Smaller matrices require far less computations per iteration and, hence, accelerate convergence of the overall spectrum.

4.2.1 Balancing a Single Matrix

The following procedure is discussed thoroughly by Wilkinson and Reinsch [112]. It seeks to create comparable row and column norms, thus reducing the Frobenius norm. We seek to find similarity transformations applied in the form

$$D_{ii}^{-1} A_{ij} D_{ii} \tag{4.5}$$

where D_{ii} is a diagonal matrix. By applying a similarity transformation of diagonal matrices the matrix norm (Frobenius) is reduced without changing the eigenvalues. To reduce rounding errors in this procedure, the diagonal elements of the similarity transformations are restricted to exact powers of the radix[†] (2 for most machines) [110]. The Matlab code shown in Alg. B.13.1 on page 321 balances a single matrix A_{ij}. It does not explicitly define D_{ii} or D_{ii}^{-1}, but rather calculates the diagonal elements of D_{ii} (assigned to the variable f and g, respectively, in the code) and applies them automatically.

4.2.2 Balancing the Generalized Eigenvalue Problem

A *regular* matrix pencil is one in which $\det(A_{ij} - \lambda B_{ij})$ is not identically zero for all values of λ. The traditional method of balancing matrix pencils is given by Ward [113]. His method attempts to make the pencil entries as close to unity as possible. It is currently the standard method for balancing in LAPACK. The method presented here is suggested by Lemonnier and Van Dooren [111]. It differs from Ward's by attempting to find a diagonal similarity transformation matrix that converts the original matrix to a *normal* matrix (or as close as possible). A normal matrix is one that commutes with its conjugate

[†]The radix (or base) is the number of unique digits, including zero, that a numeral system uses to represent numbers. For example, the decimal system has a radix of ten because it uses the ten digits 0 through 9.

transpose,[‡] $A^*_{ij}A_{ij} = A_{ij}A^*_{ij}$. This technique is fruitful since normal matrices have orthogonal eigenvectors and therefore well conditioned eigenvalues [114]. Clearly, if one can transform a poorly conditioned matrix to a normal matrix, then the calculation of the eigenvalues should be more accurate. In practice, the goal is to make a matrix *closer* to a normal matrix since an exact transformation is only realizable in exact arithmetic and cannot be accomplished with numeric precision. Lemonnier and Van Dooren suggest that this approach will consistently out-perform that by Ward [111].

As with the single matrix balancing method, we must apply appropriate similiarity transforms from both the left and the right. To define our resultant equation explicitly, we seek

$$A^{(k+1)}_{ij} - \lambda B^{(k+1)}_{ij} = {D_{ii}}_l^{-1}(A^{(k)}_{ij} - \lambda B^{(k)}_{ij})D_{iir} \tag{4.6}$$

where $D_{iil,r}$ are diagonal matrices and the l and r subscripts represent *left* and *right*, respectively [111].

It may be instructive to comment regarding the choice of $D_{iil,r}$. Any diagonal matrix applied in the form Eq. (4.5) becomes a similarity transformation that does not affect the eigenvalues. When considering a matrix pencil, we cannot simply (nor are we restricted to) applying ${D_{ii}}^{-1}(A^{(k)}_{ij} - \lambda B^{(k)}_{ij})D_{ii}$. In fact it is unlikely that $D_{iil} = D_{iir}$. The matrix D_{iil} is designed specifically to modify the rows, while D_{iir} modifies the columns. Collectively they make up the similarity transformation needed to reduce the norms while preserving the eigenvalues. By clearing parentheses, we see how to apply the similarity transforms to each matrix according to

$$A^{(k+1)}_{ij} - \lambda B^{(k+1)}_{ij} = {D_{ii}}_l^{-1}A^{(k)}_{ij}D_{iir} - \lambda {D_{ii}}_l^{-1}B^{(k)}_{ij}D_{iir} \tag{4.7}$$

[‡]A conjugate transpose, Hermitian transpose, Hermitian conjugate, or adjoint matrix, A^*_{ij}, of an $M \times N$ matrix A_{ij} with complex entries is the $N \times M$ matrix formed by taking the transpose of A_{ij} and then taking the complex conjugate of each entry.

Lastly, it should be mentioned that ${D_{ii}}_l^{-1}$ is used here rather than simply D_{iil} because, while the latter does not modify the eigenvalues, it has a tendency to increase the Frobenius norm. When dealing with matrix inversions, we should always be worried about singularity, albeit not a concern in this application.

Algorithm B.15.1 on page 327 is a modified version of the one presented by Lemonnier and Van Dooren [111].

4.2.3 Segregating Nonzero Elements

As discussed previously, grouping columns and rows with only zero elements to the top and left and constructing as many nonzero diagonal elements as possible is advantageous to increase computational speed and accuracy, especially for sparse matrices. For a matrix pencil, the B_{ij} matrix has substantial influence on the accuracy of the eigenvalues and the stability of the solver. Therefore, we can segregate nonzero elements in B_{ij} to improve results. This procedure works by first finding any columns or rows with all zero elements and pushing them to the far left and top by column and row swapping. Then the algorithm attempts to place nonzero elements on the diagonals by seeking nonzero elements above and to the left of the diagonal element in question and moving those accordingly. The same row and column swapping is applied to matrix A_{ij} to retain the correct eigenvalues. This procedure is completed before matrix reductions are applied.

Since the generalized eigenvalues are calculated by dividing after the diagonal elements of B_{ij}, this matrix takes priority in this type of algorithm. Ideally, transformations would be found that completely shift all rows and columns with only zero elements up and to the left for both constituents, A_{ij} and B_{ij}. Without significant effort exploring this possibility, this ad hoc method of preconditioning is still beneficial since the first eigenvalues computed (bottom right) since division by zero (or nearly zero) errors will not propagate into forthcoming iterations.

Algorithm B.16.1 on page 331 implements the segregation scheme on B_{ij} for a matrix pencil. This scheme is also helpful for the sparse single matrix eigenvalue problem. For a single matrix, the lines referring to matrix A_{ij} must be omitted.

4.3 Matrix Reductions

Within the limitations of numerical computations, noniterative reductions can only go so far as to convert a general complex matrix into an upper Hessenberg form. Fortuitously, practical eigensolvers iteratively reduce the elements of the lower subdiagonal of an upper Hessenberg matrix in an attempt to convert it to similar upper triangular form in which the eigenvalues appear on the diagonal. This is done by applying a series of similarity transformations that appropriately zero the subdiagonal within the bounds of numeric accuracy. For a matrix pencil, we can achieve upper Hessenberg/upper triangular form for matrices A_{ij} and B_{ij}, respectively. The application of similarity transformation matrices to the generalized eigenvalue problem at this juncture allows B_{ij} to be directly reduced to upper triangular without destroying the upper Hessenberg form of A_{ij}. The eigensolver attempts to further reduce A_{ij} to upper triangular form without destroying the upper triangular form of B_{ij} where the eigenvalues are simply a_{ii}/b_{ii}. In contrast, for a single matrix eigenvalue problem, the matrix A_{ij} can be reduced to upper triangular form directly within the eigensolver where the eigenvalues are a_{ii}.

For clarity, upper Hessenberg is a nearly triangular matrix in which all elements below the first subdiagonal are zero. It is of the form

$$\begin{bmatrix} a_{1,1} & \cdots & \cdots & a_{1,N} \\ a_{2,1} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & a_{N,N-1} & a_{N,N} \end{bmatrix} \tag{4.8}$$

Likewise, an upper triangluar matrix is of the form

$$\begin{bmatrix} b_{1,1} & \cdots & \cdots & b_{1,N} \\ 0 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & b_{N,N} \end{bmatrix} \tag{4.9}$$

4.3.1 Reduction of a Single Matrix

Three cases may be identified in which the characteristics of the original matrix can be exploited to facilitate quick computations. These are:

- A *real symmetric* matrix to tridiagonal form (special form of upper Hessenberg via Householder transformations (Alg. B.17.1 on page 333) [94].
- A *real nonsymmetric* matrix to upper Hessenberg form via Householder transformations (Alg. B.18.1 on page 336) [94].
- A *complex nonsymmetric* matrix to upper Hessenberg form via elimination techniques (Alg. B.19.1 on page 339) [110].

Clearly, the third case is the most general and encompasses the breadth of the first two. It is also the most important for the reduction of matrix pencils. Although, the first two special cases are achieved by employing Householder transformation matrices, the third uses Gaussian elimination-like row reductions with pivoting to achieve the same result. This method appears to be about twice as efficient as forming Householder similarity transformations for arbitrary complex matrices [110]. Since Gaussian elimination is not a similarity transformation and therefore changes the eigenvalues, we must apply the opposite of the row modification to the column. For instance if a subtraction operation is applied to row 2, then an equivalent addition operation must be applied to column 2. Pivoting means

finding the largest element of the column in question and moving it to the lower subdiagonal position. To ensure the similarity transform, the columns must be switched the same way. Problems can occur if zeros are on the diagonal for certain numerical routines. Pivoting can be used to ensure that these problems are avoided.

The following procedure is quoted directly from page 485 of Numerical Recipes in C [110]:

- Find the element of maximum magnitude in the rth column below the diagonal. If it is zero, skip the next two "bullets" and the stage is done. Otherwise, suppose the maximum element was in row r'
- Interchange rows r' and $r+1$. This is the pivoting procedure. To make the permutation a similarity transformation, also interchange columns r' and $r+1$
- For $i = r+2, r+3, ..., N$, compute the multiplier

$$n_{i,r+1} = \frac{a_{i,r}}{a_{r+1,r}} \tag{4.10}$$

  Subtract $n_{i,r+1}$ times row $r+1$ from row i. To make the elimination a similarity transformation, also add $n_{i,r+1}$ times column i to column $r+1$.

Note that these steps are applied $N-2$ times for the whole matrix to be reduced. We do not keep track of the transformations because we are only concerned with the eigenvalues. If the eigenvectors are required, they can be back-calculated after the eigenvalues are solved. Algorithm B.19.1 on page 339 is transcribed from the code given by Press [110] and Wilkinson [112].

4.3.2 Reduction of a Matrix Pencil

For the generalized eigenvalue problem, we must reduce a matrix pencil where A_{ij} is transformed to upper Hessenberg and B_{ij} is transformed to upper triangular. To do so, we find matrices L_{ij} and M_{ij} that, when applied as $L_{ij}A_{ij}M_{ij}$, return an upper Hessenberg matrix, while at the same time $L_{ij}B_{ij}M_{ij}$ becomes upper triangular. This is accomplished through a series of Gaussian elimination style transformations with partial pivoting - similar to Alg. B.19.1 on page 339.

This procedure, given in Kaufman's LZ algorithm [70–72], is a two step process. We begin by transforming matrix B_{ij} to upper triangular through pivoting and eliminations by finding a matrix $L_{ij}^{(0)}$ such that $L_{ij}^{(0)}B_{ij}$ is upper triangular. Since we apply $L_{ij}^{(0)}$ to B_{ij}, we must also apply it the same way to A_{ij} so that A_{ij} is replaced by $L_{ij}^{(0)}A_{ij}$. At this point, a correct form for B_{ij} emerges although A_{ij} is likely to now be incorrect.

Now we seek another transformation, $L_{ij}^{(1)}$ that zeros out A_{ij} while maintaining the triangular nature of B_{ij}. The complete similarity transformations are built one element at a time. The resulting matrix, being the product of all single element similarity transforms, can be constructed and stored, although this step is not necessary because each single element transformation can be applied directly. Keeping this point in mind: We select an element l_{N-1} in ${L_{ij}}^{(1)}$ that zeros the $(N,1)$ element in A_{ij}. When applying the transform to B_{ij}, we destroy the triangularity by adding a subdiagonal element in the $(N,N-1)$ position. Thus, after one transform, the matrices take the form

$$A_{ij} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \end{bmatrix} \quad B_{ij} = \begin{bmatrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix} \tag{4.11}$$

This action adds a nonzero subdiagonal element to B_{ij}. In order to clear that element without destroying the progress in A_{ij}, we seek a matrix M_{ij} such that $B_{ij}M_{ij}$ is returned to upper triangular form while keeping $A_{ij}M_{ij}$ unaffected. This completes the first iteration. The matrices become

$$A_{ij} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \end{bmatrix} \quad B_{ij} = \begin{bmatrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix} \tag{4.12}$$

Next, the $(N-1, 1)$ element is zeroed and B_{ij} is returned to upper triangular in the same way. The process is repeated until the reduction is complete. Kaufman calculates $13N^3/6$ multiplications and $13N^3/6$ additions for the reduction calculations. A comparable method given by Moler and Stewart [68, 69], as a precursor to their QZ algorithm, is more widely used. However, it is estimated that the matrix reductions for the QZ algorithm require $17N^3/3$ multiplications and $17N^3/3$ additions. Both methods are $\mathcal{O}(N^3)$ and therefore neither bears a distinct advantage.

These ideas are incorporated in Alg. B.20.1 on page 342

4.3.3 Block Decomposition

A matrix in upper Hessenberg form can be decomposed into smaller, nearly independent matrices if

1. An element, ϵ at the order of the machine precision is found on the subdiagonal

$$A_{ij} = \left[\begin{array}{ccc|cccc} X & X & X & X & X & X & X \\ X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X \\ \hline 0 & 0 & 0 & \epsilon & X & X & X \\ 0 & 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & 0 & X & X \end{array}\right]$$

2. Two or more moderately small elements (their product is at the order of the machine precision) are found on the subdiagonal. Decomposing on this contingency may slightly reduce the accuracy of the final result.

$$A_{ij} = \left[\begin{array}{ccc|cccc} X & X & X & X & X & X & X \\ X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X \\ \hline 0 & 0 & \epsilon^{1/2} & X & X & X & X \\ 0 & 0 & 0 & \epsilon^{1/2} & X & X & X \\ 0 & 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & 0 & X & X \end{array}\right]$$

When either case occurs, the submatrix above and below the element(s) are essentially independent and the eigensolver can be applied to the smaller submatrices independently. This often allows for reduced computational time. It also helps identify special cases in sparse matrices. For matrix pencils, decomposition is determined from A_{ij} but applied to both A_{ij} and B_{ij}.

Since this type of event is sought at each iteration of the eigensolver rather than a priori, the actual implementation is not actually preconditioning, but a convergence acceleration.

Algorithm B.21.1 on page 345 is included as a verification of the effectiveness of block decomposition since the actual implementation is a built-in feature of the eigensolver itself.

4.4 Single Matrix Eigensolvers

This section illustrates several eigensolvers of increasing sophistication. Their inclusion is specifically to identify some subtleties of simple eigensolvers that are built upon to create more sophisticated ones. Also, more sophisticated eigensolvers are often computational overkill and are less efficient for simple problems. Special cases, such as tridiagonal matrices, can be exploited to save computational time with simple, yet still accurate methods.

4.4.1 The Power Method

The power method is classified as a *deflation method* and is perhaps the simplest approach available. It does not compute matrix decompositions, nor does it require upper Hessenberg form to work; hence, it can be used for very large or sparse matrices. Unfortunately, it only computes the *largest* eigenvalue and can be very slow to converge (modifications can be made to compute other eigenvalues and to speed up the iteration). Another benefit is that the power method calculates the eigenvector automatically and can be adapted to determine the eigenvector for previously calculated eigenvalues.

The method begins by taking a guess of the dominant eigenvector x and calculating the associated Rayleigh Quotient, $\lambda = \frac{x_i' A_{ij} x_i}{x_i' x_i}$. The eigenvector is adjusted according to

$$x_{k+1} = \frac{A_{ij} x_k}{\|A_{ij} x_k\|} \tag{4.13}$$

so that x_k is multiplied by A_{ij} and normalized at each iteration. Convergence is determined when $\|A_{ij} x_i - \lambda x_i\|$ is sufficiently small.

This procedure converges under the correct assumptions that

- A_{ij} has an eigenvalue that is strictly greater in magnitude than its other eigenvalues
- The starting vector x_0 has a nonzero component in the direction of an eigenvector associated with the dominant eigenvalue

If these two assumptions hold, then x_0 will converge to the eigenvector associated with the dominant eigenvalue. This solver is listed as Alg. B.22.1 on page 347.

4.4.2 The Inverse Power Method

This method resembles the power method; however, it is known to produce faster convergence [94]. It is used to compute the eigenvalue of A_{ij} closest to a value q. If matrix A_{ij} has eigenvalues $\lambda_1, \ldots, \lambda_n$, then the eigenvalues of $(A_{ij} - qI_N)^{-1}$, where $q \neq \lambda$ are

$$\frac{1}{\lambda_1 - q}, \quad \frac{1}{\lambda_2 - q}, \quad \ldots, \quad \frac{1}{\lambda_n - q} \tag{4.14}$$

The power method is simply applied to the new formulation, $(A_{ij} - qI_N)^{-1}$.

The method converges to the eigenvalue that maximizes $1/|\lambda_k - q|$. Clearly, this means that it converges to the eigenvalue closest to q in order to minimize the denominator, thus leading to the largest fraction. The guess, q, is known as a shift. It is used to accelerate convergence in nearly every eigensolver. Although the implementation of a shift in this method is fairly simple (i.e. q does not need to be updated), more advanced eigensolvers will update the shift at every iteration in order to accelerate convergence further. Ideally, q should be close to an eigenvalue of A_{ij} to minimize the number of iterations. The closer it is to an eigenvalue, the faster the convergence. Therefore, a good estimate of q can be obtained from the Geršgorin Circle Theorem[§] or from Raleigh's Quotient. Both approaches yield sufficiently accurate a priori estimates. For eigensolvers that update the shift at each

[§]On a number line, the center of the circle is taken to be the diagonal element a_{ii}. The radius of the circle is the sum of the absolute value of the off-diagonal elements in the ith row. The corresponding eigenvalue will lie within one radius of a_{ii}.

iteration, it is also acceptable to start with the eigenvalues of submatrix $A^{(k)}_{N-1,N-1}$ as a initializing guess [109]. Furthermore, selecting the 2×2 submatrix in the lower right corner is beneficial for eigensolvers that return the entire spectrum since it calculates two eigenvalues and can reduce the overall size of A_{ij} via block decompositions.

The application of the Inverse Power Method is implemented in Alg. B.23.1 on page 349. For the interested reader, Burden and Faires [94] provide a concise and easily followed discussion of this method.

4.4.3 The QR and LR Methods

Deflation methods are straightforward, but they tend to be impractical in calculating the whole spectrum if the matrix is singular, or computational time is important. They can also succumb to round-off error due to the larger number of iterations required for convergence. The *QR Method* uses orthogonal similarity transformations devised from *QR decompositions* to calculate all the eigenvalues simultaneously by converging to an upper triangular matrix where the eigenvalues are on the diagonal [94]. This decomposition was first implemented by Francis to solve single matrix eigenvalue problems [115, 116]. In brief, the QR decomposition sets

$$A_{ij} = Q_{ij} R_{ij} \tag{4.15}$$

where Q_{ij} is an orthogonal matrix and R_{ij} is upper triangular. In general terms, it follows a simple two-step process

1. $A^{(1)}_{ij} = A_{ij}$ is factored as $Q^{(1)}_{ij} R^{(1)}_{ij}$
2. $A^{(2)}_{ij}$ is defined as $A^{(2)}_{ij} = R^{(1)}_{ij} Q^{(1)}_{ij}$

The process is iterated until $A^{(k)}_{ij}$ becomes upper triangular and the eigenvalues appear on the diagonal.

In this case, the details are important in understanding how the method works. Assuming we can determine the QR decomposition of $A_{ij}^{(k)}$ where $Q_{ij}^{(k)}$ is orthogonal and $A_{ij}^{(k+1)} = R_{ij}^{(k)} Q_{ij}^{(k)}$, then $R_{ij}^{(k)} = Q_{ij}^{(k)^t} A_{ij}^{(k)}$. In this event we can write

$$A_{ij}^{(k+1)} = R_{ij}^{(k)} Q_{ij}^{(k)} = (Q_{ij}^{(k)^t} A_{ij}^{(k)}) Q_{ij}^{(k)} = Q_{ij}^{(k)^t} A_{ij}^{(k)} Q_{ij}^{(k)} \tag{4.16}$$

The eigenvalues of $A_{ij}^{(k+1)}$ are equal to those of A_{ij} because $A_{ij}^{(k)}$ is only multiplied by orthogonal matrices.

For further explanation of the QR method with simple examples see Burden and Faires [94]. Special types, such as symmetric tridiagonal matrices, can be exploited for accelerated convergence. In general though, the matrix A_{ij} must at least be upper Hessenberg.

The LR decomposition method is a comparable approach. Similar to the QR method, it seeks transformation matrices that will zero the lower subdiagonal. Where both methods iteratively build the transformation matrix by zeroing one element at a time, the QR uses orthogonal matrices whereas the LR uses eliminations with each iteration multiplying the product of all previous transformations. The method is originally based on the observations by Rutishauser [117] that, much like the QR decomposition if

$$A_{ij} = L_{ij} R_{ij} \tag{4.17}$$

where L_{ij} is a lower triangular matrix and R_{ij} is upper triangular. Therefore,

$$A_{ij}^{(k)} = L_{ij}^{(k)} R_{ij}^{(k)}; \quad A_{ij}^{(k+1)} = R_{ij}^{(k)} L_{ij}^{(k)} \tag{4.18}$$

where, as with the QR method, $R_{ij}^{(k)} = L_{ij}^{(k)^{-1}} A_{ij}^{(k)}$. Thus,

$$A_{ij}^{(k+1)} = L_{ij}^{(k)^{-1}} A_{ij}^{(k)} L_{ij}^{(k)} \tag{4.19}$$

is similar to $A_{ij}^{(k)}$.

Recall from linear algebra, that the inverse of an orthogonal matrix is its transpose (i.e. if Q is an orthogonal matrix, then $Q^t = Q^{-1}$). We could have easily written Eq. (4.16) as $A_{ij}^{(k+1)} = {Q_{ij}^{(k)}}^{-1} A_{ij}^{(k)} Q_{ij}^{(k)}$: a form nearly identical to that of Eq. (4.19). The fundamental difference between the two decompositions is that L_{ij} is not required to be orthogonal. This allow the freedom to construct it with more elimination-type operations. It remains a similarity transformation as long as it is paired with its inverse in Eq. (4.19); ensuring the eigenvalues remain unchanged.

It has been shown that this iterative sequence tends toward upper triangular form given that A_{ij} has roots of distinct modulii (absolute magnitude), i.e. when the diagonal elements are the roots arranged in order of decreasing modulus [117, 118]. Furthermore, it has been suggested that the LR has a slightly faster convergence time than the QR [71]. A more general version of these solvers, capable of handling complex matrices, are dubbed the QZ and LZ methods, respectively. Two in-house versions of the QZ and LZ algorithms are found in Algs. B.24.1 on page 352 and B.25.1 on page 361, respectively.

4.5 Generalized Eigensolvers

Both the QZ and LZ decompositions can be adapted to solve the generalized eigenvalue problem. Moler and Stewart are responsible for the advent of the generalized QZ algorithm [68, 69], while the LZ is attributed to Kaufman [70–72]. The latter states that the LZ-algorithm is based on three observations:

1. If L_{ij} and M_{ij} are nonsingular matrices, the eigenvalue problem $L_{ij}A_{ij}M_{ij}y_i = \lambda L_{ij}B_{ij}M_{ij}y_i$ and $A_{ij}x_i = \lambda B_{ij}x_i$ have the same eigenvalues and their eigenvectors are related by $x_i = M_{ij}y_i$.

2. If A_{ij} and B_{ij} are triangular matrices with diagonal elements α_i and β_i, then for $i = 1, 2, \ldots, N$, α_i/β_i are eigenvalues of the generalized eigenvalue problem if $\beta_i \neq 0$. If $\alpha_i \neq 0$ and $\beta_i = 0$, then infinity is an eigenvalue. If both $\alpha_i = \beta_i = 0$, then all scalars are eigenvalues.

3. There exists matrices L_{ij} and M_{ij} such that $L_{ij}A_{ij}M_{ij}$ and $L_{ij}B_{ij}M_{ij}$ are both upper triangular and L_{ij} and M_{ij} are products of lower triangular and permutation matrices.

Both the QZ and LZ algorithms hinge on similar precepts. In the QZ algorithm, the matrices L_{ij} and M_{ij} are orthogonal, while in the LZ they are products of stabilized elementary transformations [71].

The implementation of the LZ algorithm is summarized as the iterative process:

1. Find $L_{ij}^{(0)}$ and $M_{ij}^{(0)}$ such that $A_{ij}^{(1)} = L_{ij}^{(0)}A_{ij}M_{ij}^{(0)}$ is upper Hessenberg and $B_{ij}^{(1)} = L_{ij}^{(0)}B_{ij}M_{ij}^{(0)}$ is upper triangular (see Reduction of a Matrix Pencil on page 138).

2. Iteratively reduce A_{ij} to upper triangular while preserving the triangularity of B_{ij}. The iterations include:

   - Find a shift appropriate for the current eigenvalue, s.
   - Find matrices $L_{ij}^{(k)}$ and $M_{ij}^{(k)}$ such that $L_{ij}^{(k)}(A_{ij}^{(k)} - \lambda_k B_{ij}^{(k)})$ and $L_{ij}^{(k)}B_{ij}^{(k)}M_{ij}^{(k)}$ are upper triangular.
   - Set $A_{ij}^{(k+1)} = L_{ij}^{(k)}A_{ij}^{(k)}M_{ij}^{(k)}$ and $B_{ij}^{(k+1)} = L_{ij}^{(k)}B_{ij}^{(k)}M_{ij}^{(k)}$. A_{ij} will be reverted back to upper Hessenberg form with a diminished lower subdiagonal.

The shift can be estimated in a number of ways as previously discussed. To understand how the matrices L_{ij} and M_{ij} are constructed, we start with upper Hessenberg/upper triangular matrices A_{ij} and B_{ij}, respectively, and apply $L_{ij}^{(1)}$ to both results in matrices

of the form

$$L_{ij}^{(1)} A_{ij}^{(1)} = \begin{bmatrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix} \quad L_{ij}^{(1)} B_{ij}^{(1)} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix} \tag{4.20}$$

Now, $M_{ij}^{(1)}$ is applied to return B_{ij} back to upper triangular. Applying this transformation to A_{ij}, returns it to upper Hessenberg form.

$$A_{ij}^{(2)} = L_{ij}^{(1)} A_{ij}^{(1)} M_{ij}^{(1)} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix} \quad B_{ij}^{(2)} = L_{ij}^{(1)} B_{ij}^{(1)} M_{ij}^{(1)} = \begin{bmatrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix} \tag{4.21}$$

$L_{ij}^{(2)}$ is selected to annihilate element (3,2) in $A_{ij}^{(2)}$. Applying $L_{ij}^{(2)}$ to $B_{ij}^{(2)}$ introduces a nonzero subdiagonal element in the (3,2) position:

$$L_{ij}^{(2)} A_{ij}^{(2)} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix} \quad L_{ij}^{(2)} B_{ij}^{(2)} = \begin{bmatrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix} \tag{4.22}$$

This new nonzero element in B_{ij} is eliminated by $M_{ij}^{(2)}$ and returns A_{ij} to upper Hessenberg:

$$A_{ij}^{(3)} = L_{ij}^{(2)} A_{ij}^{(2)} M_{ij}^{(2)} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix} \quad B_{ij}^{(3)} = L_{ij}^{(2)} B_{ij}^{(2)} M_{ij}^{(2)} = \begin{bmatrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix} \tag{4.23}$$

This is called a "bulge chasing" scheme since the "bulge" created below the diagonal is pushed down the matrix. This is iterated until the bulge is eliminated completely and the matrices are returned to upper Hessenberg/upper triangular form. The complete transformation matrices take the form

$$L_{ij} = L_{ij}^{(N)} L_{ij}^{(N-1)} L_{ij}^{(N-2)} \dots L_{ij}^{(1)} \quad \text{and} \quad M_{ij} = M_{ij}^{(1)} M_{ij}^{(2)} M_{ij}^{(3)} \dots M_{ij}^{(N)} \tag{4.24}$$

Since each iteration operates on specific rows and columns matrices A_{ij} and B_{ij}, it is not necessary to store the completed transformation matrices at each iteration. Doing so can unnecessarily cost computer resources and should be avoided. Upon completion of the bulge chasing scheme, A_{ij} should be returned back to upper Hessenberg with significantly diminished subdiagonal elements. This illustration applies equally well to the single matrix eigenvalue problem. Remember, in that case, B_{ij} exists but is simply the identity matrix. The process is iterated and new transformation matrices are constructed until the subdiagonal elements of A_{ij} are sufficiently close to zero and A_{ij} can be treated as upper triangular. In this form, the eigenvalues are given as $\lambda_i = a_{ii}/b_{ii}$.

Since the rate of convergence of the subdiagonal elements to zero is fastest from the bottom right corner to top left corner, the last row and column can be ignored in subsequent

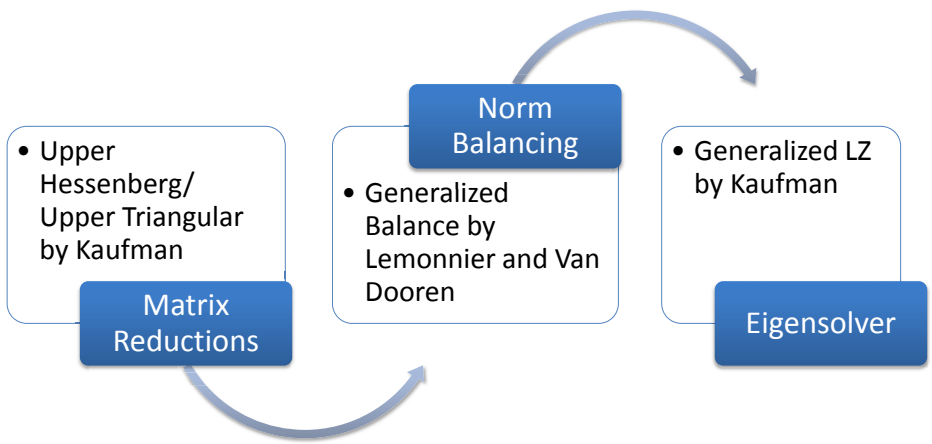


Figure 4.2: Eigensolver Flowchart

calculations once the $(N, N-1)$ subdiagonal is sufficiently small. This reduces the overall size of the matrix and accelerates the computation of the remaining eigenvalues.

4.5.1 Example: Implementation of an LZ Eigensolver

For a eigensolver flowchart including algorithm options at each step we refer the reader to Fig. 4.1. Specifically, we implement a complete eigensolver according to Fig. 4.2. These selections are based on efficiency and availability of the respective algorithms.

As an example, we consider the eigenvalue problem given by the matrices[¶]:

$$A_{ij} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix} \quad B_{ij} = \begin{bmatrix} 4 & -6 & 4 & -1 \\ -6 & 14 & -11 & 3 \\ 4 & -11 & 10 & -3 \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{4.25}$$

[¶]Collectively, these matrices have no physical significance. A_{ij} is a "magic" matrix with equal row and column sums and B_{ij} is an inverse symmetric Pascal matrix.

The first step is to convert to upper Hessenberg/upper triangular form. We follow the algorithm presented by Kaufman [70, 71] and found in Alg. B.20.1 on page 342. This results in matrices of the form:

$$A_{ij} = \begin{bmatrix} 5.00 & 17.75 & 17.85 & 10.00 \\ 206.0 & 562.2 & 671.9 & 403.0 \\ 0.000 & -3.838 & -17.22 & -11.46 \\ 0.000 & 0.000 & -3.352 & -2.391 \end{bmatrix} \quad B_{ij} = \begin{bmatrix} -6.000 & 1.015 & 6.471 & -11.00 \\ 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.159 & 2.667 \\ 0.000 & 0.000 & 0.000 & 1.050 \end{bmatrix} \tag{4.26}$$

Next, matrix balancing is applied to ensure accurate results. To do so, we implement the algorithm by Lemmonier and Van Dooren [111] as coded in Alg. B.15.1 on page 327. Before applying the algorithm, the Frobenius norms are calculated to be $\|A_{ij}\| = 986.7$ and $\|B_{ij}\| = 14.46$

$$A_{ij} = \begin{bmatrix} 0.625 & 0.555 & 0.279 & 0.313 \\ 0.805 & 0.549 & 0.328 & 0.394 \\ 0.000 & -0.240 & -0.538 & -0.716 \\ 0.000 & 0.000 & -0.419 & -0.598 \end{bmatrix} \quad B_{ij} = \begin{bmatrix} -0.750 & 0.032 & 0.101 & -0.344 \\ 0.000 & 0.001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.005 & 0.167 \\ 0.000 & 0.000 & 0.000 & 0.263 \end{bmatrix} \tag{4.27}$$

Now the norms are reduced to $\|B_{ij}\| = 1.865$ and $\|B_{ij}\| = 0.888$: a significant reduction.

There are no zero elements to be segregated so this step is unnecessary. Rather, we begin the eigensolver by calculating and applying an acceleration shift estimated by the eigenvalue

of $A_{N-1,N-1:N,N}$. Applying the shift to A_{ij} results in the matrices:

$$A_{ij}^{(1)} = \begin{bmatrix} 0.407 & 0.564 & 0.308 & 0.213 \\ 0.805 & 0.549 & 0.328 & 0.394 \\ 0.000 & -0.240 & -0.537 & -0.668 \\ 0.000 & 0.000 & -0.419 & -0.521 \end{bmatrix} \quad B_{ij}^{(1)} = \begin{bmatrix} -0.750 & 0.032 & 0.101 & -0.344 \\ 0.000 & 0.001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.005 & 0.167 \\ 0.000 & 0.000 & 0.000 & 0.263 \end{bmatrix} \tag{4.28}$$

$B_{ij}^{(1)}$ is unmodified from B_{ij}.

The transformation matrices L_{ij} and M_{ij} are constructed a column at a time. This leads to the bulge chasing scheme described above. For clarity, we will illustrate one complete formation of the two transformation matrices. We first find $L_{ij}^{(1)}$ to eliminate element (2,1). We find,

$$L_{ij}^{(1)} A_{ij}^{(1)} = \begin{bmatrix} 0.407 & 0.564 & 0.308 & 0.213 \\ 0.000 & -0.565 & -0.281 & -0.027 \\ 0.000 & -0.240 & -0.537 & -0.668 \\ 0.000 & 0.000 & -0.419 & -0.521 \end{bmatrix}$$

$$L_{ij}^{(1)} B_{ij}^{(1)} = \begin{bmatrix} -0.750 & 0.032 & 0.101 & -0.344 \\ 1.481 & -0.062 & -0.200 & 0.679 \\ 0.000 & 0.000 & 0.005 & 0.167 \\ 0.000 & 0.000 & 0.000 & 0.263 \end{bmatrix} \tag{4.29}$$

The (2,1) element in A_{ij} is annihilated at the expense of adding a subdiagonal element to B_{ij} in the (2,1) position. B_{ij} is returned to upper triangular form through the transformation

matrix $M_{ij}^{(1)}$. This operation leaves:

$$A_{ij}^{(2)} = L_{ij}^{(1)} A_{ij}^{(1)} M_{ij}^{(1)} = \begin{bmatrix} 13.96 & 0.564 & 0.308 & 0.213 \\ -13.57 & -0.565 & -0.281 & -0.027 \\ 0.000 & -0.240 & -0.537 & -0.668 \\ 0.000 & 0.000 & -0.419 & -0.521 \end{bmatrix}$$

$$B_{ij}^{(2)} = L_{ij}^{(1)} B_{ij}^{(1)} M_{ij}^{(1)} = \begin{bmatrix} 0.012 & 0.032 & 0.101 & -0.344 \\ 0.000 & -0.062 & -0.200 & 0.679 \\ 0.000 & 0.000 & 0.005 & 0.167 \\ 0.000 & 0.000 & 0.000 & 0.263 \end{bmatrix} \tag{4.30}$$

Now the subdiagonal element (3,2) is eliminated from $A_{ij}^{(2)}$. We have:

$$L_{ij}^{(2)} A_{ij}^{(2)} = \begin{bmatrix} 13.96 & 0.564 & 0.308 & 0.213 \\ -13.57 & -0.565 & -0.281 & -0.027 \\ 0.000 & 0.000 & -0.418 & -0.657 \\ 0.000 & 0.000 & -0.419 & -0.521 \end{bmatrix}$$

$$L_{ij}^{(2)} B_{ij}^{(2)} = \begin{bmatrix} 0.012 & 0.032 & 0.101 & -0.344 \\ 0.000 & -0.062 & -0.200 & 0.679 \\ 0.000 & 0.026 & 0.090 & -0.122 \\ 0.000 & 0.000 & 0.000 & 0.263 \end{bmatrix} \tag{4.31}$$

Again, M_{ij} is used to restore B_{ij} to its original form:

$$A_{ij}^{(3)} = L_{ij}^{(2)} A_{ij}^{(2)} M_{ij}^{(2)} = \begin{bmatrix} 13.96 & 0.474 & 0.308 & 0.213 \\ -13.57 & -0.483 & -0.281 & -0.027 \\ 0.000 & 0.122 & -0.418 & -0.657 \\ 0.000 & 0.000 & -0.419 & -0.521 \end{bmatrix}$$

$$B_{ij}^{(3)} = L_{ij}^{(2)} B_{ij}^{(2)} M_{ij}^{(2)} = \begin{bmatrix} 0.012 & 0.002 & 0.101 & -0.344 \\ 0.000 & -0.003 & -0.200 & 0.679 \\ 0.000 & 0.000 & 0.090 & -0.122 \\ 0.000 & 0.000 & 0.000 & 0.263 \end{bmatrix} \tag{4.32}$$

Finally, we attempt to eliminate the last subdiagonal element in A_{ij} in the (4,3) position. This results in:

$$L_{ij}^{(3)} A_{ij}^{(3)} = \begin{bmatrix} 13.96 & 0.474 & 0.308 & 0.213 \\ -13.57 & -0.483 & -0.281 & -0.027 \\ 0.000 & 0.122 & -0.418 & -0.657 \\ 0.000 & 0.000 & 0.000 & 0.137 \end{bmatrix}$$

$$L_{ij}^{(3)} B_{ij}^{(3)} = \begin{bmatrix} 0.012 & 0.002 & 0.101 & -0.344 \\ 0.000 & -0.003 & -0.200 & 0.679 \\ 0.000 & 0.000 & 0.090 & -0.122 \\ 0.000 & 0.000 & -0.090 & 0.385 \end{bmatrix} \tag{4.33}$$

Once again, M_{ij} completes the iteration by returning B_{ij} to upper triangular form:

$$A_{ij}^{(4)} = L_{ij}^{(3)} A_{ij}^{(3)} M_{ij}^{(3)} = \begin{bmatrix} 13.96 & 0.474 & 0.358 & 0.213 \\ -13.57 & -0.483 & -0.287 & -0.027 \\ 0.000 & 0.122 & -0.571 & -0.657 \\ 0.000 & 0.000 & 0.032 & 0.137 \end{bmatrix}$$

$$B_{ij}^{(4)} = L_{ij}^{(3)} B_{ij}^{(3)} M_{ij}^{(3)} = \begin{bmatrix} 0.012 & 0.002 & 0.021 & -0.344 \\ 0.000 & -0.003 & -0.041 & 0.679 \\ 0.000 & 0.000 & 0.061 & -0.122 \\ 0.000 & 0.000 & 0.000 & 0.385 \end{bmatrix} \tag{4.34}$$

This completes the formation of transformation matrices L_{ij} and M_{ij}. At this point, little progress appears to be made toward our initial goal of reducing A_{ij} to upper triangular form. However, applying this process again recovers:

$$A_{ij}^{(4)} = \begin{bmatrix} 18.34 & 0.429 & 0.356 & 0.305 \\ -0.199 & -0.030 & 0.068 & 0.088 \\ 0.000 & 0.027 & -0.212 & -0.129 \\ 0.000 & 0.000 & 1.8\text{E-}4 & 0.014 \end{bmatrix} \quad B_{ij}^{(4)} = \begin{bmatrix} 0.032 & 0.000 & 0.016 & -0.344 \\ 0.000 & 0.001 & -0.016 & 0.345 \\ 0.000 & 0.000 & -0.032 & 1.826 \\ 0.000 & 0.000 & 0.000 & 0.664 \end{bmatrix} \tag{4.35}$$

We see that $A_{4,3}$ is now much smaller than at the end of the first iteration and $A_{3,2}$ is also beginning to diminish. After another iteration, we have

$$A_{ij}^{(4)} = \begin{bmatrix} 18.23 & 0.439 & 0.356 & 0.312 \\ 0.007 & -0.023 & 0.072 & 0.084 \\ 0.000 & -0.004 & -0.136 & -0.080 \\ 0.000 & 0.000 & 3.8\text{E-}9 & 0.000 \end{bmatrix} \quad B_{ij}^{(4)} = \begin{bmatrix} 0.033 & 0.000 & 0.016 & -0.344 \\ 0.000 & 0.001 & -0.016 & 0.341 \\ 0.000 & 0.000 & -0.049 & 2.186 \\ 0.000 & 0.000 & 0.000 & 0.667 \end{bmatrix} \tag{4.36}$$

Now, $A_{4,3}$ is orders of magnitude smaller and $A_{3,2}$ is also smaller with this iteration. Applying the scheme again finds

$$A_{ij}^{(4)} = \begin{bmatrix} 18.23 & 0.438 & 0.356 & 0.313 \\ -3.0\text{E-}4 & -0.024 & 0.072 & 0.083 \\ 0.000 & 4.9\text{E-}4 & -0.148 & -0.095 \\ 0.000 & 0.000 & -3.1\text{E-}19 & 0.000 \end{bmatrix} \quad B_{ij}^{(4)} = \begin{bmatrix} 0.033 & 0.000 & 0.016 & -0.344 \\ 0.000 & 0.001 & -0.016 & 0.341 \\ 0.000 & 0.000 & -0.046 & 2.128 \\ 0.000 & 0.000 & 0.000 & 0.667 \end{bmatrix} \tag{4.37}$$

After this iteration, we see all the subdiagonal elements in A_{ij} are quickly approaching zero and $A_{4,3}$ has fallen beyond numeric zero. Thus, we can focus the forthcoming calculations on the submatrix $A_{1:3,1:3}$. Also, B_{ij} is no longer changing by any appreciable amount. Another iteration gives

$$A_{ij}^{(4)} = \begin{bmatrix} 18.13 & 0.438 & 0.305 & 1.398 \\ 1.5\text{E-}5 & -0.027 & 0.123 & -0.994 \\ 0.000 & 2.7\text{E-}10 & 0.000 & -6.811 \\ 0.000 & 0.000 & -3.1\text{E-}19 & -2.105 \end{bmatrix} \quad B_{ij}^{(4)} = \begin{bmatrix} 0.033 & 0.000 & 0.016 & -0.344 \\ 0.000 & 0.001 & -0.016 & 0.341 \\ 0.000 & 0.000 & -0.046 & 2.128 \\ 0.000 & 0.000 & 0.000 & 0.667 \end{bmatrix} \tag{4.38}$$

Iterating again finds

$$A_{ij}^{(4)} = \begin{bmatrix} 18.13 & 0.438 & 0.305 & 1.398 \\ -7.7\text{E-}7 & -0.027 & 0.123 & -0.994 \\ 0.000 & -4.1\text{E-}24 & 0.000 & -6.811 \\ 0.000 & 0.000 & -3.1\text{E-}19 & -2.105 \end{bmatrix} \quad B_{ij}^{(4)} = \begin{bmatrix} 0.033 & 0.000 & 0.016 & -0.344 \\ 0.000 & 0.001 & -0.016 & 0.341 \\ 0.000 & 0.000 & -0.046 & 2.128 \\ 0.000 & 0.000 & 0.000 & 0.667 \end{bmatrix} \tag{4.39}$$

Table 4.1: Eigenvalues computed for the given example compared to those computed via Lapack.

Calculated	Lapack	Difference
560.808144698150000	560.808144698153000	2.95586E-12
-24.965140100619800	-24.965140100619600	1.98952E-13
3.1569954024660500	3.1569954024660500	0
0.0000000000000000	0.0000000000000002	1.99839E-16

Now $A_{3,2}$ is sufficiently small and the submatrix can be reduced again by one row and one column. Applying a final iteration completes the reduction. We have,

$$A_{ij}^{(4)} = \begin{bmatrix} 19.04 & 0.438 & 0.305 & 1.398 \\ 4.1\text{E-}24 & -0.027 & 0.123 & -0.994 \\ 0.000 & -4.1\text{E-}24 & 0.000 & -6.811 \\ 0.000 & 0.000 & -3.1\text{E-}19 & -2.105 \end{bmatrix} \quad B_{ij}^{(4)} = \begin{bmatrix} 0.033 & 0.000 & 0.016 & -0.344 \\ 0.000 & 0.001 & -0.016 & 0.341 \\ 0.000 & 0.000 & -0.046 & 2.128 \\ 0.000 & 0.000 & 0.000 & 0.667 \end{bmatrix} \tag{4.40}$$

The eigenvalues for this problem are given by computing a_{ii}/b_{ii}. They are listed in Table 4.1. As we can see, the results are corroborated by standard Lapack routines using the similar QZ algorithm built into MATLAB with the largest error associated with the largest eigenvalue. This example is computed by Alg. B.26.1 on page 376.

4.6 Closing Remarks on Eigensolvers

The preceding chapter stands to illuminate the framework of an efficient and accurate eigensolver. These techniques are not new but the need to fully understand their implementation has fallen waywardly with the advent of Lapack, Arpack, Blas, and other linear algebra packages. In practice, it is more efficient to rely on these packaged codes to solve large scale eigenvalue problems. The same is true here. Although capable of handling stability problems, the algorithms discussed here have larger time requirements then the

optimized codes in Lapack. Therefore, for the sake of computational time, we too utilized the Lapack functions within MATLAB, though we now understand their framework and implementation.

Another important eigensolver has gone unmentioned. The Arnoldi eigensolver and its derivatives have become accepted as the defacto method for stability analysis. It can accommodate sparse matrices with efficient computations. It also resolves only a prespecified number of eigenvalues closest to a given start value. This approach certainly appears advantageous; however, it may not be so. Numeric experimentation shows convergence of the Arnoldi solver in MATLAB is not significantly faster for large matrices. Compounded by the necessity to iterate over several initial eigenvalues to capture the whole spectrum, the total time to resolve the problem may actually be much longer. These statements are true for the pseudo-sparse matrices generated by the stability equations discussed in the forthcoming chapters. For dense matrices with a large number of nonzero/noninfinite eigenvalues, the QZ/LZ algorithms appear to have an advantage. For extremely sparse large matrices with very few nonzero/noninfinite eigenvalues, the Arnoldi algorithms may be worth considering. For stability analysis, the selection of eigensolver is subjective to the formulation. Arnoldi methods are more accommodating for the matrix size and density of streamfunction formulations where the entire problem is reduced to a single dependent variable. In reality, a general stability formulation will disallow the introduction of the streamfunction and expand the size of the matrices by a factor of sixteen. At this point, Arnoldi is no longer worthwhile to pursue.

One advantage of the Arnoldi algorithm is that it will compute a predefined number of eigenvalues. Since the smallest eigenvalues are the first calculated by the LZ algorithm (assuming no significant block decompositions are available), and recalling that these eigenvalues are the most physically accurate, the LZ algorithm can simply be interrupted after a satisfactory set of eigenvalues has been collected. This is mentioned in passing only and not recommended without closer examination of the actual upper Hessenberg/upper

triangular form of the hydrodynamic eigenvalue problem. Its spectral form will be subject to the formulation and if significant block decompositions are available, interrupting the eigensolver may be ill-advised.

Chapter 5

Local Nonparallel Stability Analysis of the Bidirectional Vortex

To date, the study by Abu-Irshaid, Majdalani and Casalis is the only attempt to characterize the hydrodynamic instability of the bidirectional vortex [119]. Their efforts were applied to a local nonparallel stability analysis and resulted in some interesting findings. Unfortunately, the validity of their analysis comes under three questions. The first being the physical validity of a one-dimensional, parallel flow analysis for a strongly two-dimensional flow, the second coming in the form of typographical errors in the original document undermining the confidence of the study, and the third refers to the inclusion of three-dimensional boundary layers along the chamber wall. Regardless of the correctness of the original study, it is necessary to regenerate this work for comparison purposes.

5.1 Deriving the Spectral LNP Equations

The stability analysis begins by considering the *Linearized Navier-Stokes* equations (LNS) appropriate for the *local nonparallel* (LNP) stability analysis. The complete derivation is found in App. A.1. Here we present the results directly:

Continuity:

$$\frac{\mathrm{d}u_r}{\mathrm{d}r} + \frac{u_r}{r} + iq\frac{u_\theta}{r} + i\alpha u_z = 0 \tag{5.1a}$$

Radial momentum:

$$\begin{aligned} -i\omega u_r + U_r\frac{\mathrm{d}u_r}{\mathrm{d}r} + u_r\frac{\partial U_r}{\partial r} + iq\frac{U_\theta u_r}{r} + \frac{u_\theta}{r}\frac{\partial U_r}{\partial \theta} - 2\frac{U_\theta u_\theta}{r} + i\alpha U_z u_r + u_z\frac{\partial U_r}{\partial z} + \frac{\mathrm{d}p}{\mathrm{d}r} \\ = \varepsilon\left[\frac{\mathrm{d}^2 u_r}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}u_r}{\mathrm{d}r} - \frac{u_r}{r^2} - q^2\frac{u_r}{r^2} - 2iq\frac{u_\theta}{r^2} - \alpha^2 u_r\right] \end{aligned} \tag{5.1b}$$

Tangential momentum:

$$\begin{aligned} -i\omega u_\theta + U_r\frac{\mathrm{d}u_\theta}{\mathrm{d}r} + u_r\frac{\partial U_\theta}{\partial r} + iq\frac{U_\theta u_\theta}{r} + \frac{u_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r u_\theta}{r} + \frac{u_r U_\theta}{r} + i\alpha U_z u_\theta + u_z\frac{\partial U_\theta}{\partial z} + iq\frac{p}{r} \\ = \varepsilon\left[\frac{\mathrm{d}^2 u_\theta}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}u_\theta}{\mathrm{d}r} - \frac{u_\theta}{r^2} - q^2\frac{u_\theta}{r^2} + 2iq\frac{u_r}{r^2} - \alpha^2 u_\theta\right] \end{aligned} \tag{5.1c}$$

Axial momentum:

$$\begin{aligned} -i\omega u_z + U_r\frac{\mathrm{d}u_z}{\mathrm{d}r} + u_r\frac{\partial U_z}{\partial r} + iq\frac{U_\theta u_z}{r} + \frac{u_\theta}{r}\frac{\partial U_z}{\partial \theta} + i\alpha U_z u_z + u_z\frac{\partial U_z}{\partial z} + i\alpha p \\ = \varepsilon\left[\frac{\mathrm{d}^2 u_z}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}u_z}{\mathrm{d}r} - q^2\frac{u_z}{r^2} - \alpha^2 u_z\right] \end{aligned} \tag{5.1d}$$

We see that our governing equations lend themselves to an eigenvalue problem containing a system of ODEs. To formulate and solve this problem we will refer to Sec. 3.4.3 and Sec. 3.5.1 as a guide.

Rather than two equations as in our example, we have four to contend with. The procedure is the same; however, the implementation can easily lead to mistakes if care is not given. Since we still seek to obtain the form of the generalized eigenvalue problem,

$$A_{ij} f_i = \lambda B_{ij} f_i \tag{5.2}$$

we must build the operator matrices from the governing system. To do so we build both operator matrices from the smaller $N \times N$ block matrices such that A_{ij} and B_{ij} take the form

$$A_{ij} = \begin{array}{r} \\ \text{Cont.} \rightarrow \\ r-\text{mom.} \rightarrow \\ \theta-\text{mom.} \rightarrow \\ z-\text{mom.} \rightarrow \end{array} \begin{array}{c} \begin{array}{cccc} u_r(r) & u_\theta(r) & u_z(r) & p(r) \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \left[\begin{array}{c|c|c|c} A_{c,u_r} & A_{c,u_\theta} & A_{c,u_z} & A_{c,p} \\ \hline A_{r,u_r} & A_{r,u_\theta} & A_{r,u_z} & A_{r,p} \\ \hline A_{\theta,u_r} & A_{\theta,u_\theta} & A_{\theta,u_z} & A_{\theta,p} \\ \hline A_{z,u_r} & A_{z,u_\theta} & A_{z,u_z} & A_{z,p} \end{array}\right] \end{array}$$

$$B_{ij} = \begin{array}{r} \\ \text{Cont.} \rightarrow \\ r-\text{mom.} \rightarrow \\ \theta-\text{mom.} \rightarrow \\ z-\text{mom.} \rightarrow \end{array} \left[\begin{array}{c|c|c|c} B_{c,u_r} & B_{c,u_\theta} & B_{c,u_z} & B_{c,p} \\ \hline B_{r,u_r} & B_{r,u_\theta} & B_{r,u_z} & B_{r,p} \\ \hline B_{\theta,u_r} & B_{\theta,u_\theta} & B_{\theta,u_z} & B_{\theta,p} \\ \hline B_{z,u_r} & B_{z,u_\theta} & B_{z,u_z} & B_{z,p} \end{array}\right]$$

where each row is constructed from the equation referenced on the left and each column from the operators acting on the dependent variable given at the top. The final matrices are $4N \times 4N$.

First we rewrite the governing equations in operator form as usual. They become

Continuity:

$$\left(\frac{\mathrm{d}}{\mathrm{d}r}+r^{-1}\right)u_r+\left(iqr^{-1}\right)u_\theta+(ia)u_z=0 \tag{5.3a}$$

Radial momentum:

$$\begin{aligned}\left\{U_r\frac{\mathrm{d}}{\mathrm{d}r}+\frac{\partial U_r}{\partial r}+iqU_\theta r^{-1}+i\alpha U_z-\varepsilon\left[\frac{\mathrm{d}^2}{\mathrm{d}r^2}+r^{-1}\frac{\mathrm{d}}{\mathrm{d}r}-\left(1+q^2\right)r^{-2}-\alpha^2\right]\right\}u_r\\ +\left(2iq\varepsilon r^{-2}-2U_\theta r^{-1}+r^{-1}\frac{\partial U_r}{\partial\theta}\right)u_\theta+\left(\frac{\partial U_r}{\partial z}\right)u_z+\left(\frac{\mathrm{d}}{\mathrm{d}r}\right)p=(i\omega)\,u_r\end{aligned} \tag{5.3b}$$

Tangential momentum:

$$\begin{aligned}\left(\frac{\partial U_\theta}{\partial r}+U_\theta r^{-1}-2iq\varepsilon r^{-2}\right)u_r+\left\{U_r\frac{\mathrm{d}}{\mathrm{d}r}+U_r r^{-1}+iqU_\theta r^{-1}+r^{-1}\frac{\partial U_\theta}{\partial\theta}+i\alpha U_z\right.\\ \left.-\varepsilon\left[\frac{\mathrm{d}^2}{\mathrm{d}r^2}+r^{-1}\frac{\mathrm{d}}{\mathrm{d}r}-\left(1+q^2\right)r^{-2}-\alpha^2\right]\right\}u_\theta+\left(\frac{\partial U_\theta}{\partial z}\right)u_z+\left(iqr^{-1}\right)p=(i\omega)\,u_\theta\end{aligned} \tag{5.3c}$$

Axial momentum:

$$\begin{aligned}\left(\frac{\partial U_z}{\partial r}\right)u_r+\left(r^{-1}\frac{\partial U_z}{\partial\theta}\right)u_\theta+\left[U_r\frac{\mathrm{d}}{\mathrm{d}r}+iqU_\theta r^{-1}+\frac{\partial U_z}{\partial z}+i\alpha U_z\right.\\ \left.-\varepsilon\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2}+r^{-1}\frac{\mathrm{d}}{\mathrm{d}r}-q^2r^{-2}-\alpha^2\right)\right]u_z+(i\alpha)\,p=(i\omega)\,u_z\end{aligned} \tag{5.3d}$$

Our domain of interest is $0\le r\le 1$, thus we require the mapping

$$r=\frac{1}{2}(\xi+1)\quad\longleftrightarrow\quad\xi=2r-1\qquad\text{likewise,}\qquad\frac{\mathrm{d}}{\mathrm{d}x}=2\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{5.4}$$

Following the procedure previously laid out, we can define the block matrices from the operator form of the governing equations. They are

$$\left\{\begin{array}{lcl} A_{c,u_r} & = & \bar{D}_N + r_{ii}^{-1} \\ A_{c,u_\theta} & = & iqr_{ii}^{-1} \\ A_{c,u_z} & = & i\alpha I_N \\ A_{c,p} & = & 0 \end{array}\right. \quad \text{and} \left\{\begin{array}{lcl} B_{c,u_r} & = & 0 \\ B_{c,u_\theta} & = & 0 \\ B_{c,u_z} & = & 0 \\ B_{c,p} & = & 0 \end{array}\right. \tag{5.5}$$

$$\left\{\begin{array}{lcl} A_{r,u_r} & = & U_{rii}\bar{D}_N + \left(\dfrac{\partial U_r}{\partial r}\right)_{ii} + iqU_{\theta ii}r_{ii}^{-1} + i\alpha U_{zii} \\ & & -\varepsilon\left[(\bar{D}_N)^2 + r_{ii}^{-1}\bar{D}_N - (1+q^2)\,r_{ii}^{-2} - \alpha^2 I_N\right] \\ A_{r,u_\theta} & = & 2iq\varepsilon r_{ii}^{-2} - 2U_{\theta ii}r_{ii}^{-1} + r_{ii}^{-1}\left(\dfrac{\partial U_r}{\partial \theta}\right)_{ii} \\ A_{r,u_z} & = & \left(\dfrac{\partial U_r}{\partial z}\right)_{ii} \\ A_{r,p} & = & \bar{D}_N \end{array}\right. \quad \text{and} \left\{\begin{array}{lcl} B_{r,u_r} & = & iI_N \\ B_{r,u_\theta} & = & 0 \\ B_{r,u_z} & = & 0 \\ B_{r,p} & = & 0 \end{array}\right. \tag{5.6}$$

$$\left\{\begin{array}{lcl} A_{\theta,u_r} & = & \left(\dfrac{\partial U_\theta}{\partial r}\right)_{ii} + U_{\theta ii}r_{ii}^{-1} - 2iq\varepsilon r_{ii}^{-2} \\ A_{\theta,u_\theta} & = & U_{rii}\bar{D}_N + U_{rii}r_{ii}^{-1} + iqU_{\theta ii}r_{ii}^{-1} \\ & & +r_{ii}^{-1}\left(\dfrac{\partial U_\theta}{\partial \theta}\right)_{ii} + i\alpha U_{zii} \\ & & -\varepsilon\left[(\bar{D}_N)^2 + r_{ii}^{-1}\bar{D}_N - (1+q^2)\,r_{ii}^{-2} - \alpha^2 I_N\right] \\ A_{\theta,u_z} & = & \left(\dfrac{\partial U_\theta}{\partial z}\right)_{ii} \\ A_{\theta,p} & = & iqr_{ii}^{-1} \end{array}\right. \quad \text{and} \left\{\begin{array}{lcl} B_{\theta,u_r} & = & 0 \\ B_{\theta,u_\theta} & = & iI_N \\ B_{\theta,u_z} & = & 0 \\ B_{\theta,p} & = & 0 \end{array}\right. \tag{5.7}$$

$$
\begin{cases}
A_{z,u_r} &= \left(\dfrac{\partial U_z}{\partial r}\right)_{ii} \\
A_{z,u_\theta} &= r_{ii}^{-1}\left(\dfrac{\partial U_z}{\partial \theta}\right)_{ii} \\
A_{z,u_z} &= U_{rii}\bar{D}_N + iqU_{\theta ii}r_{ii}^{-1} + \left(\dfrac{\partial U_z}{\partial z}\right)_{ii} + i\alpha U_{zii} \\
& \quad -\varepsilon\left[(\bar{D}_N)^2 + r_{ii}^{-1}\bar{D}_N - q^2 r_{ii}^{-2} - \alpha^2 I_N\right] \\
A_{z,p} &= i\alpha I_N
\end{cases}
\quad \text{and} \quad
\begin{cases}
B_{z,u_r} &= 0 \\
B_{z,u_\theta} &= 0 \\
B_{z,u_z} &= iI_N \\
B_{z,p} &= 0
\end{cases}
\tag{5.8}
$$

where ω is the eigenvalue represented as λ in the generalized eigenvalue equation, $A_{ij}f_i = \lambda B_{ij}f_i$. Spectral methods can also be used to compute the derivatives of the base flow with respect to r while derivatives of with respect to θ and z must be defined specifically.

To remain as general as possible we must define two sets of boundary conditions: one for the case of axisymmetric perturbations ($q = 0$) and the second for all other cases ($q = 1, 2, 3, ...$). At all physical boundaries we rely on the standard acoustic boundary conditions $\boldsymbol{n} \cdot \boldsymbol{u} = 0$ and $\boldsymbol{n} \cdot \nabla p = 0$. For axisymmetric perturbations, the boundary conditions are determined to be

$$
\begin{bmatrix}
(I_N)_{N,:} & (0) & (0) & (0) \\
(I_N)_{1,:} & (0) & (0) & (0) \\
(0) & (I_N)_{N,:} & (0) & (0) \\
(0) & (I_N)_{1,:} & (0) & (0) \\
(0) & (0) & (\bar{D}_N)_{N,:} & (0) \\
(0) & (0) & (I_N)_{1,:} & (0) \\
(0) & (0) & (0) & (\bar{D}_N)_{N,:} \\
(0) & (0) & (0) & (D_N)_{1,:}
\end{bmatrix}
\begin{bmatrix}
u_r(0) \\ u_r(1) \\ u_\theta(0) \\ u_\theta(1) \\ u_z(0) \\ u_z(1) \\ p(0) \\ p(1)
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}
\tag{5.9}
$$

Likewise, for nonsymmetric perturbations they are

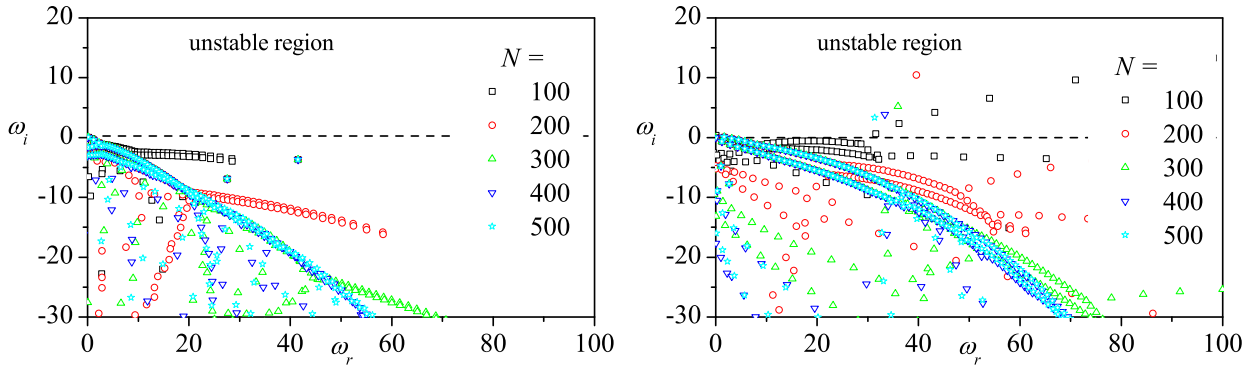
$$\begin{bmatrix} (I_N)_{N,:} & (0) & (0) & (0) \\ (I_N)_{1,:} & (0) & (0) & (0) \\ (0) & (I_N)_{N,:} & (0) & (0) \\ (0) & (I_N)_{1,:} & (0) & (0) \\ (0) & (0) & \boldsymbol{(I_N)_{N,:}} & (0) \\ (0) & (0) & (I_N)_{1,:} & (0) \\ (0) & (0) & (0) & \boldsymbol{(I_N)_{N,:}} \\ (0) & (0) & (0) & (D_N)_{1,:} \end{bmatrix} \begin{bmatrix} u_r(0) \\ u_r(1) \\ u_\theta(0) \\ u_\theta(1) \\ u_z(0) \\ u_z(1) \\ p(0) \\ p(1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.10}$$

The difference here (denoted in Eq. (5.10) by bold typeface) is the exclusion of derivative boundary conditions at the centerline for the axial wave and pressure perturbation. In any event, the difference in the results is found to be small.

5.2 Code Validation and Grid Refinement

The spectral solutions presented here are derived and coded completely independent of the original study. They follow the conventions formally laid out in Sec. 3. Figures 5.1a–5.1b generate results for input values cited by Abu-Irshaid, Majdalani, and Casalis to validate the correctness of our code. Despite limitations* in their paper, the values calculated here overlay well with those in the original study [119]; however, without their tabulated results, the exact degree of agreement is unknown. Initial test runs verify the original solution is correct within the context of the LNP approach for the inviscid (semi-inviscid) base flow and our new code reproduces those results almost identically.

*In their paper, they apply the LNP approach to a short chamber in which the parallel flow assumption appears to be unsuitable.

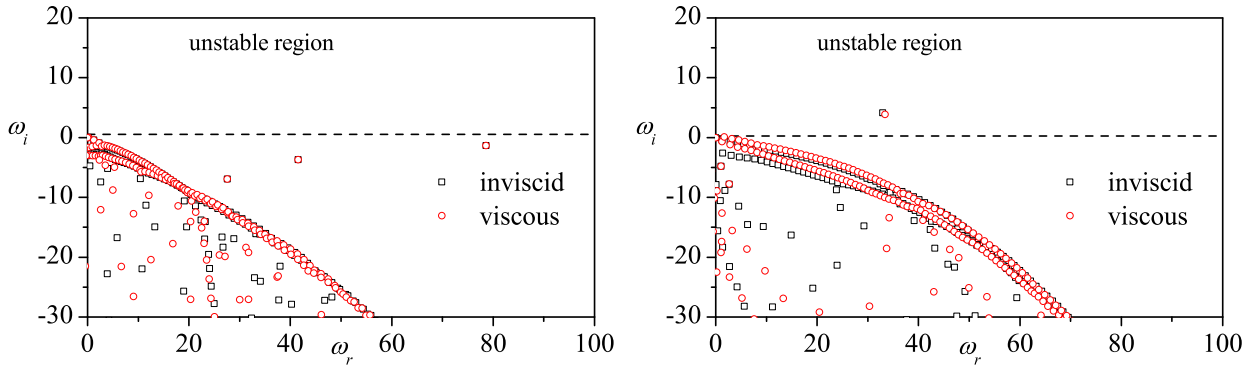


a) Axisymmetric perturbations ($q = 0$) **b)** Asymmetric perturbations ($q = 1$)

Figure 5.1: Grid refinement for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

The previous study suggests that convergence occurs for $N = 600$. Using the present code, Fig. 5.1 shows that satisfactory convergence can be expected for $N \geq 400$ over the interval in question. Recall from Table 3.2, that $N/3$ eigenvalues were accurate to 5×10^{-4}. If this observation holds, we can expect $100 - 150$ eigenvalues to share this degree of accuracy. This is corroborated by the distinct, noncontinuous line of eigenvalues defining a boundary above which few eigenvalues exist. Convergence of eigenvalues below this boundary is indicated by clustering of eigenvalues with successive increases in N. It is probable that further resolution of these eigenvalues is beyond the limitations of current computational methods either in terms of machine precision [95] or computing power. Either way, convergence of these eigenvalues is inconsequential since they do not grow in time.

The original study applies the stability analysis to a semi-inviscid base flow. The inclusion of the tangential forced core vortex eliminated the singularity emanating from the r^{-1} inviscid solution; however their study foes not consider sidewall boundary layers. In Figs. 5.2a–5.2b, we compare the spectrum for both under similar conditions. The insignificant disparity for the majority of the spectrum suggests that the inclusion of boundary layers at the sidewall does not have a large overall effect. It does, however, produce several unstable modes near $[0, 0]$ that are not present in the inviscid solution. Thus we can conclude, that



a) Axisymmetric perturbations ($q = 0$) **b)** Asymmetric perturbations ($q = 1$)

Figure 5.2: Comparison of the spectrum for the inviscid versus the viscous solution with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

while the overall effect is not large, the importance of viscosity in the base flow with regards to the generation of unstable modes may be. The nature of parietal vortex shedding is consistent with the generation of low frequency unstable waves caused by from the wall/boundary layer interactions. It may speculated that the bulk inviscid flow breakdown is inherently stable; at least for this set of input parameters. Increasing the Reynolds number, the primary controlling parameter of the boundary layer character, shows a larger disparity between the inviscid and viscous solutions for low frequency eigenvalues.

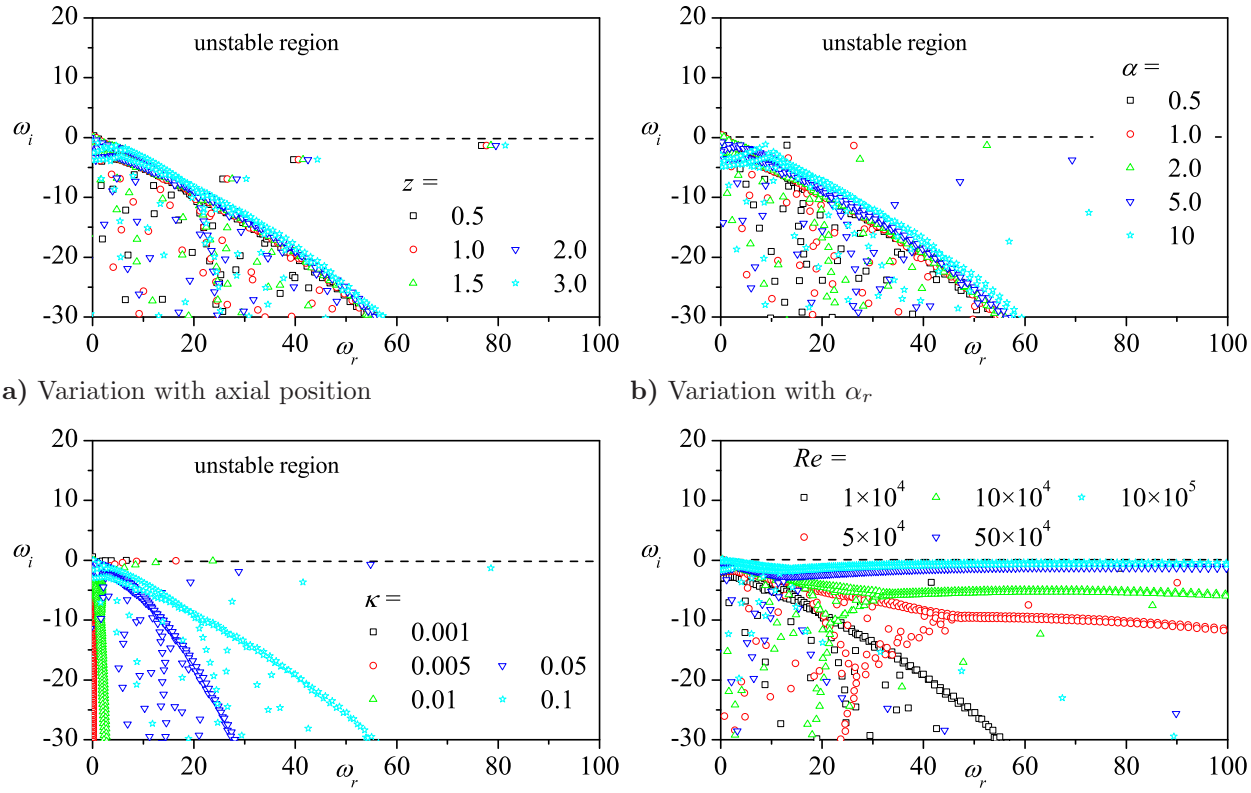
For comparison purposes we take $z = 1.5$ whenever possible. Argument could be made against this choice but this axial position represents a midpoint between the regions where the parallel flow assumption is blatantly violated and where it is apparently met (see Fig. 1.3).

5.3 The Complex-Lamellar Bidirectional Vortex

As previously mentioned, the results presented here are in close agreement with those by Abu-Irshaid, Majdalani, and Casalis [119]. In this section we reproduce those results as well as identify the axisymmetric and asymmetric hydrodynamic wave form corresponding to a specific case.

5.3.1 Axisymmetric Spectrum

Figures 5.3a–5.3d present a parametric study of the axisymmetric spectrum for four key parameters. We continue to see coherent structures present in all four subfigures. It is apparent that if amplified modes are present, they most often occur at very low frequencies. The near-continuous spectrum of higher frequency eigenvalues remain essentially damped for the majority of input parameters. Some interesting results are present in Fig. 5.3a. The results for each axial position examined are nearly the same. This is substantiated by a nearly identical pseudo-continuous spectral line. This includes an offshoot of the primary spectral line back toward the zero axis. This structure appears to asymptote to the $\omega_i = 0$ axis but, according to our results, does not cross it. This asymptotic behavior appears in other studies as well. Clearly, the effect of axial position and wave number are much less influential than that of κ or Re. For both parameters we see drastic changes in the spectrum including a larger number of amplified eigenvalues. Interestingly enough, increasing κ flattens the spectral line but does not necessarily increase the number of amplified eigenvalues. By increasing κ, the swirl intensity is lowered. Thus, the trend toward greater damping of the eigenvalues corresponds to a decrease in the swirl intensity of the base flow. In fact, for $\kappa \leq 0.005$, the spectrum is nearly vertical indicating very small circular frequency coupled with very high damping. This result could be attributed to greater centrifugal forces inhibiting the onset of vortex shedding. It becomes apparent that the excitation of specific eigenvalues is very parameter dependent. Figure 5.3d suggests that very high injection Reynolds numbers excite an increasingly larger number of eigenvalues. This is an expected result. It should be noted, however, that even though both κ and Re are characteristic parameters of the viscous layers, it is there effect on the base flow and LNP equations (not on the boundary layer profile) that promotes the most pronounced effect on the spectrum and not boundary layer properties.



a) Variation with axial position

b) Variation with α_r

c) Variation with κ

d) Variation with Reynolds number

Figure 5.3: Axisymmetric parametric study for several input parameters. Here, $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

5.3.2 Asymmetric Spectrum

It comes into question the importance of including asymmetric effects in the hydrodynamic stability equations. It has been shown that for spatial stability the earliest onset of instability is attributed to axisymmetric modes for Taylor plane and Taylor-Culick flow [55]. For truly parallel flows the $q = 0$ condition is proven to be sufficient by Squire's Theorem [120]. Gaster shows this theorem holds for three dimensional inviscid flows [121] but could not extend it further. Here, however, we include viscosity in both the base flow and the stability equations. Also, the bidirectional vortex flows are three-dimensional and result in a significantly different

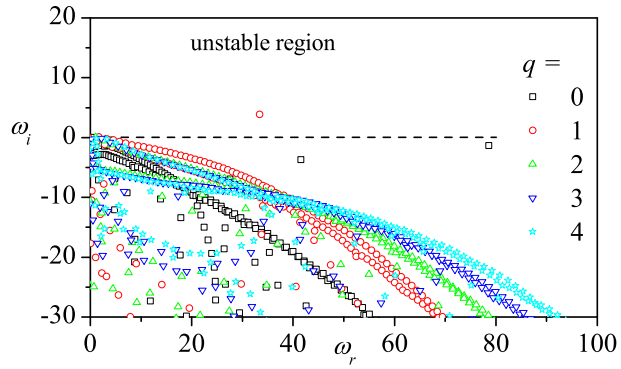
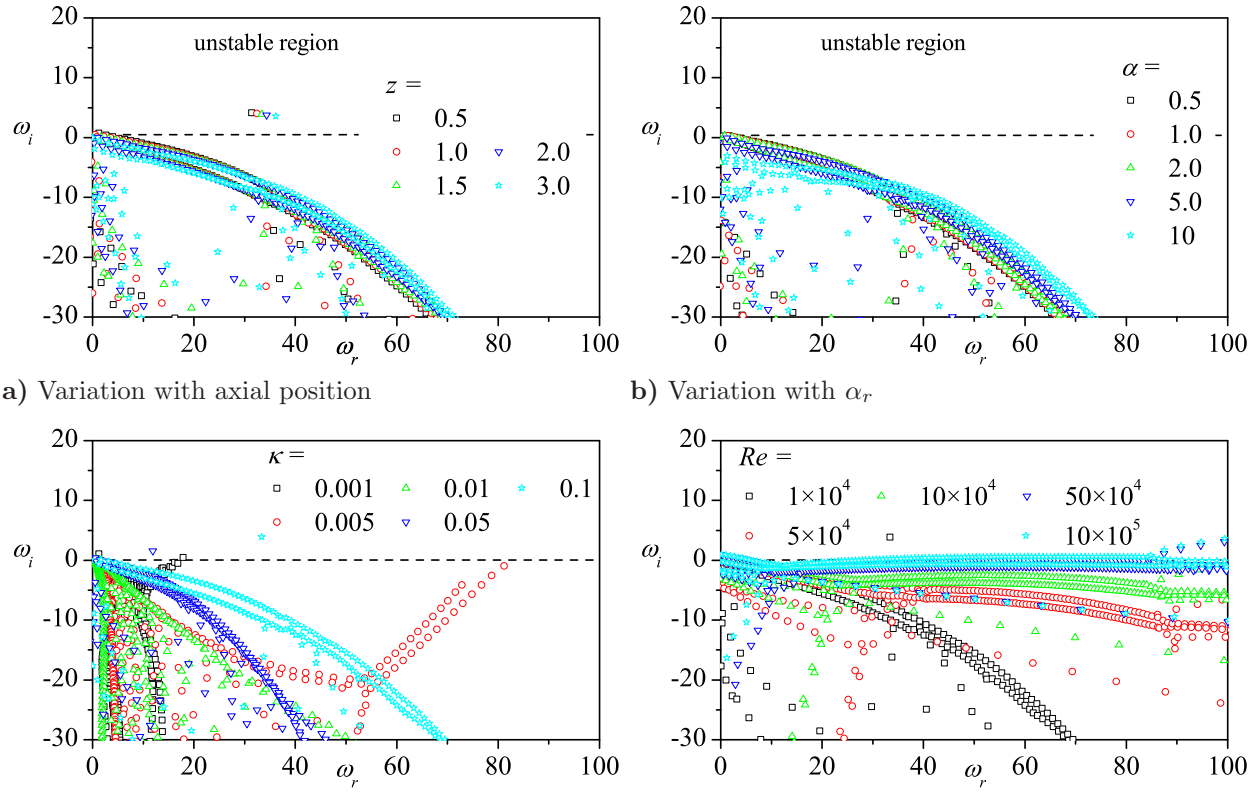


Figure 5.4: The effect of higher tangential mode numbers with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

spectral decomposition than the closely related Taylor-Culick flow. The inclusion of a third viscous dimension in the base flow requires a study of asymmetric instability.

Figure 5.4 suggests that higher mode numbers do affect the temporal spectrum by flattening the spectral line but do not necessarily have a large effect on unstable modes. This flattening effect increases with increasing q with the largest value plotted here possessing a saddle shape. The amplified eigenvalue around $\omega = 35 + 4i$ is only present for the $q = 1$ case with $\alpha = 3$. It is not present in Fig. 5.5b nor anywhere else but is still present when all other parameters are varied. This single point could be an extraneous result, but its persistence throughout the study suggests otherwise. It reinforces the sensitivity of instability on initial parameters.

The parametric study of the asymmetric spectrum in Fig. 5.5 once again show the greatest effects originating from variations κ and Re. Both these parameters show abrupt bifurcations from the pseudo-continuous spectral lines. For high Reynolds numbers, this translates to greater amplification of high frequency eigenvalues than previously seen. It can be shown in Fig. 5.6 that the circular frequency, ω_r, is linearly dependent on axial position. For instance, plotting the real part of the undamped eigenvalue near $\omega = 35 + 4i$ versus axial



a) Variation with axial position

b) Variation with α_r

c) Variation with κ

d) Variation with Reynolds number

Figure 5.5: Asymmetric parametric study for several input parameters. Here $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

position generates a nearly straight line with a positive slope. Likewise, the imaginary part shows a nearly straight line with negative slope.

5.3.3 Multiple Mantles

Multiple mantle solutions show a greater number of unstable modes for increasing mantle number, m, in Fig. 5.7. Both show a flattening in the spectral lines with the most drastic effect seen in the asymmetric case. We also observe a vertical and lateral stretch in the position of subsequent eigenvalues along the pseudo-continuous spectral line.

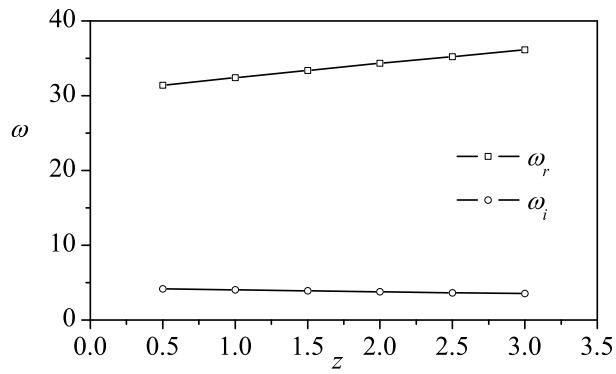
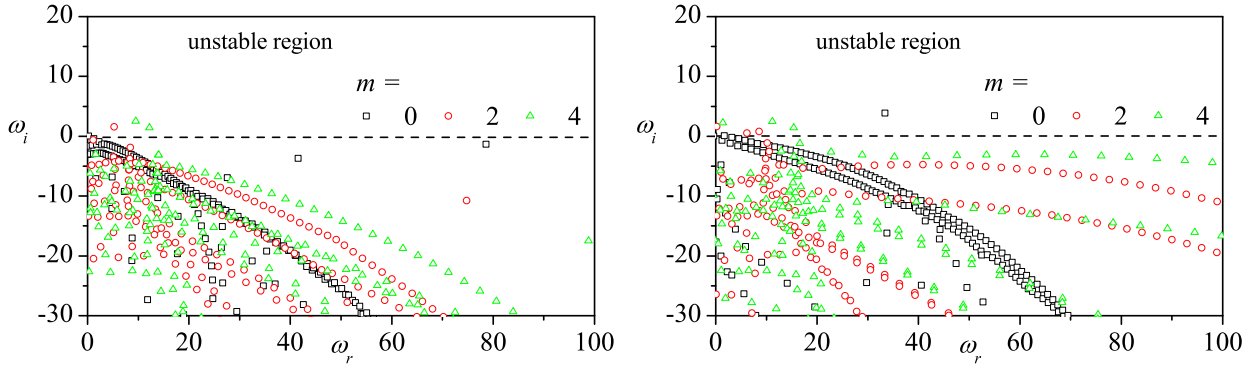


Figure 5.6: Illustrating the nearly linear axial dependence of the spectrum for the undamped eigenvalue near $\omega = 35 + 4i$ with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.



a) Axisymmetric perturbations ($q = 0$) **b)** Asymmetric perturbations ($q = 1$)

Figure 5.7: The effect of multiple mantles on the temporal spectrum for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

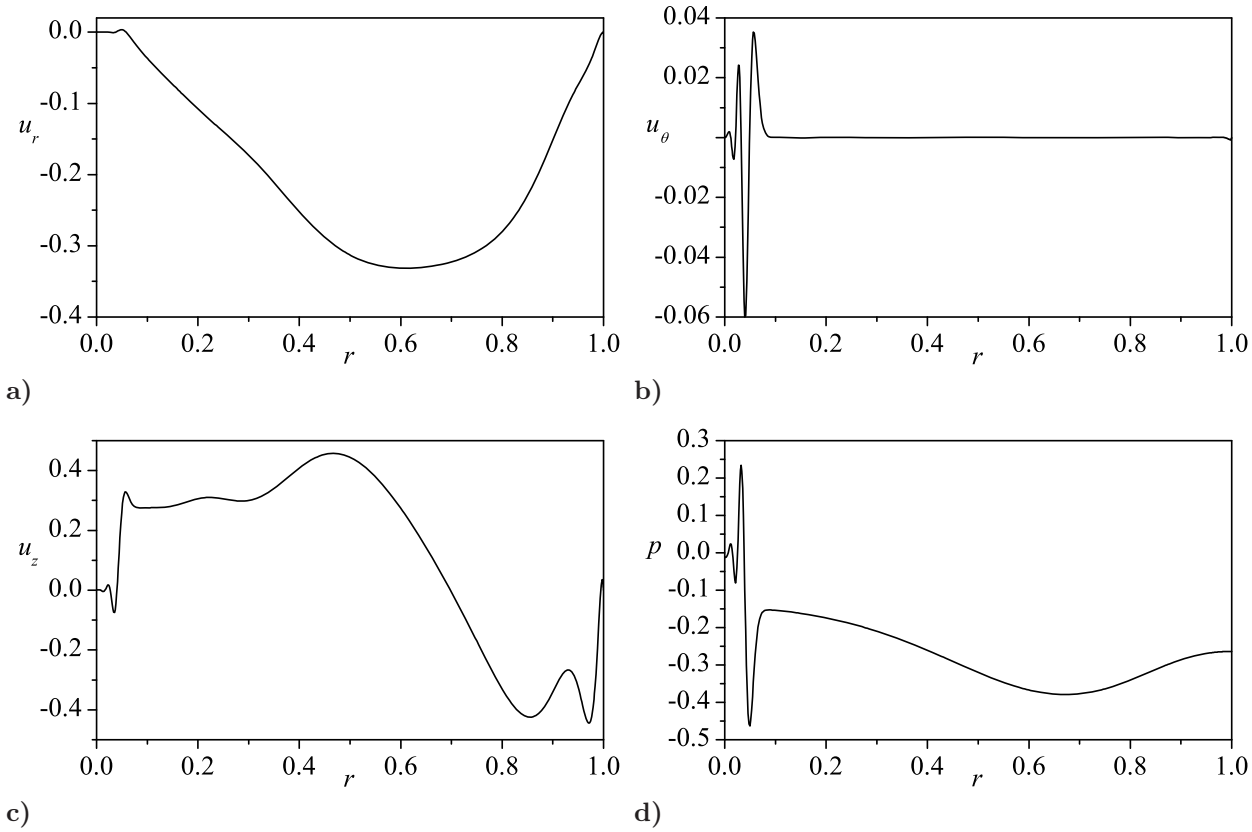
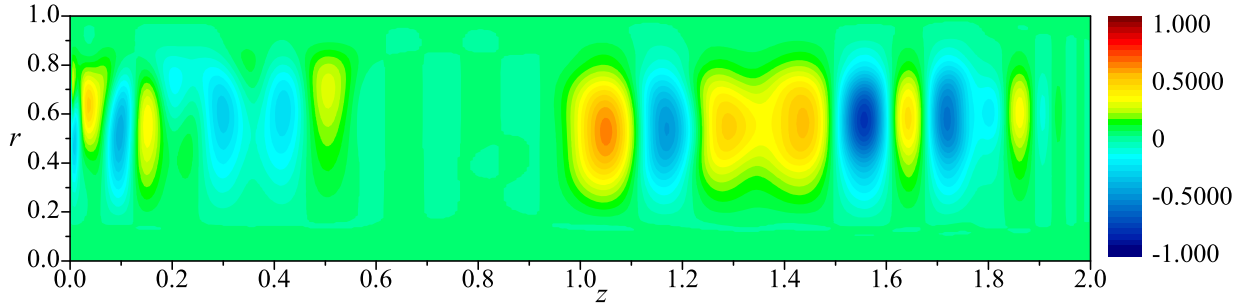


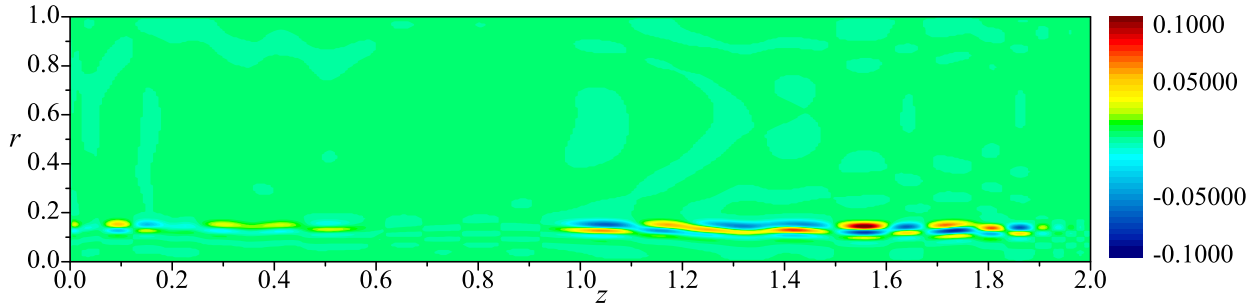
Figure 5.8: Axisymmetric eigenvectors for the first undamped frequency, $\omega = 0.1911 + 0.0625i$, with the input parameters $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values.

5.3.4 Amplified Frequencies

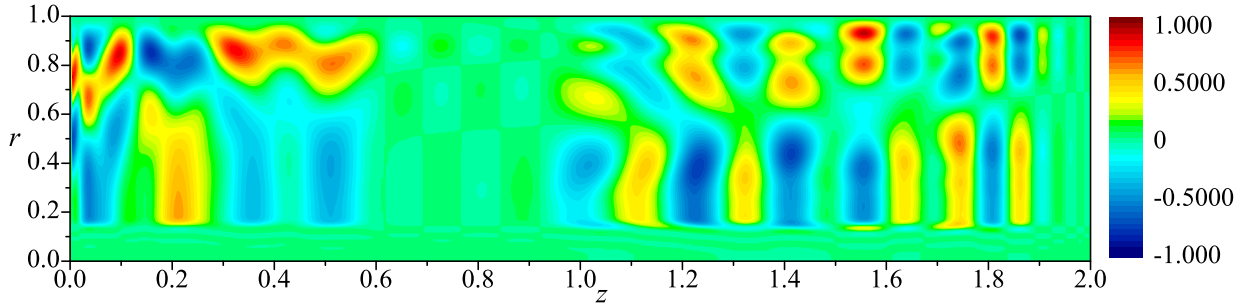
Hydrodynamic wave forms come in a variety of shapes. Figures 5.8–5.10 show two examples. Close attention should be paid to the axial, tangential, and pressure profile. Both the axisymmetric and asymmetric cases show a cohesive wave form near the wall with slight disparity near the centerline. The exact cause of this is unknown. It could be inherent in the retention of r^{-1} terms in the LNP equations forcing the interval of analysis to be [**eps**,1] where **eps**=`2.2204E-016` is the distance from 1.0 to the next largest double-precision number. Clearing r^{-1} from the governing equations produces similar but significantly less accurate results. In the paper by French and Majdalani [122], this technique was used to resolve



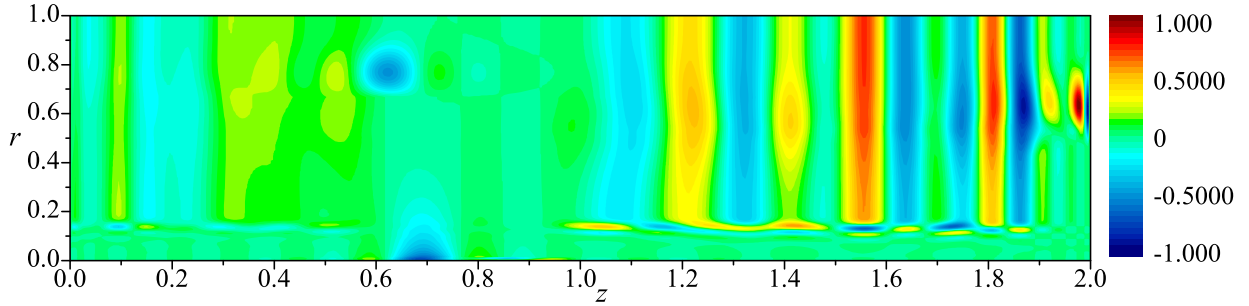
a) LNP axisymmetric radial velocity wave for the complex-lamellar model



b) LNP axisymmetric tangential velocity wave for the complex-lamellar model



c) LNP axisymmetric axial velocity wave for the complex-lamellar model



d) LNP axisymmetric pressure wave for the complex-lamellar model

Figure 5.9: Axisymmetric contour plots, with $q = 0$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$.

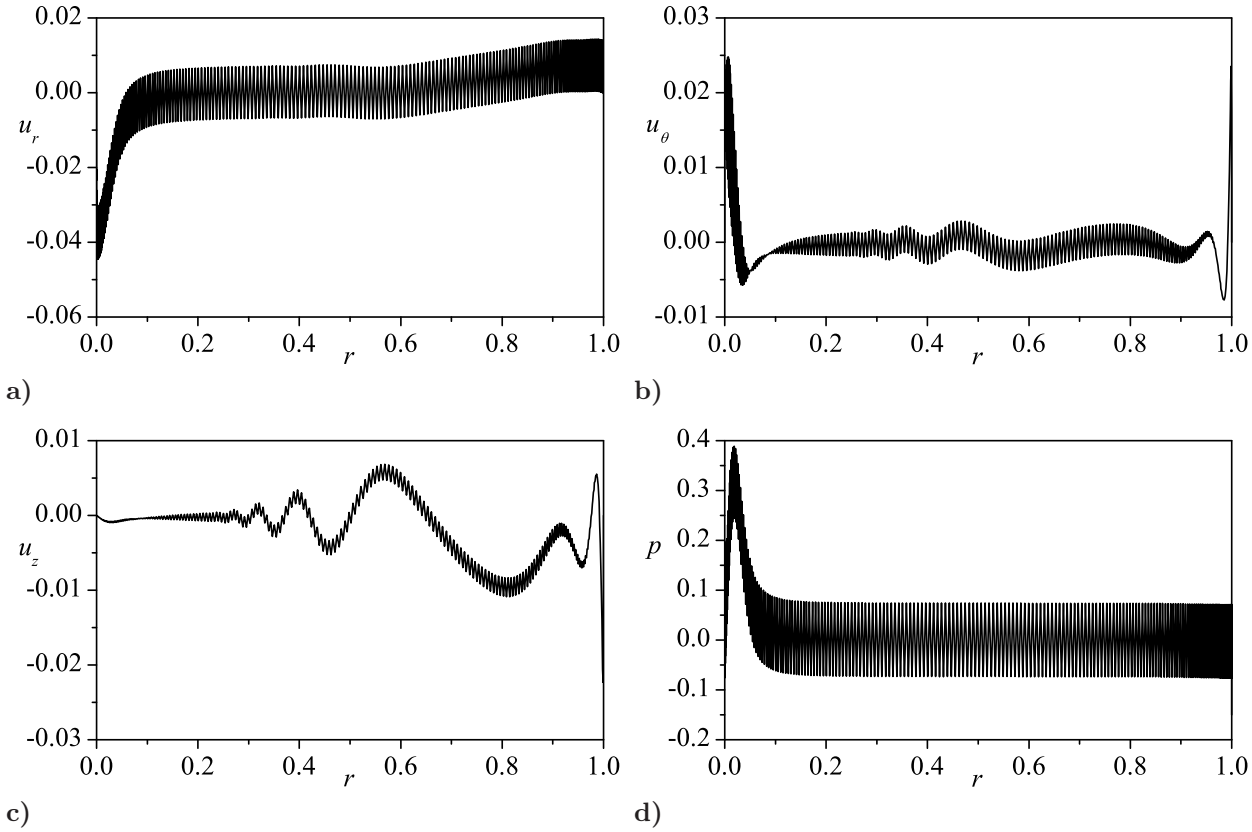
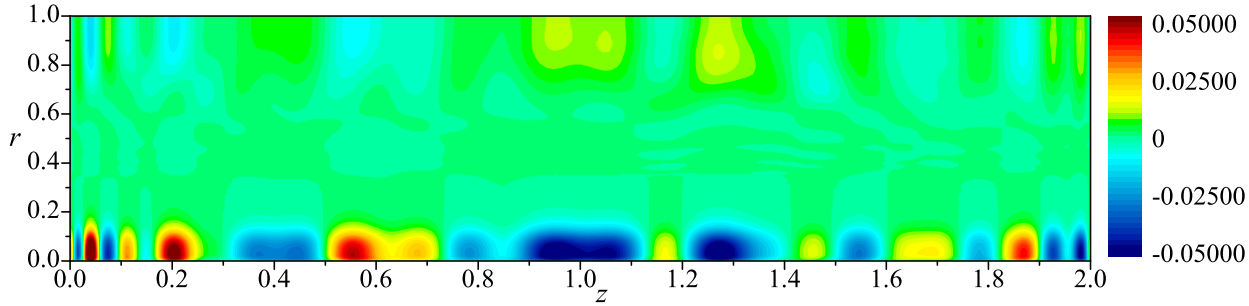
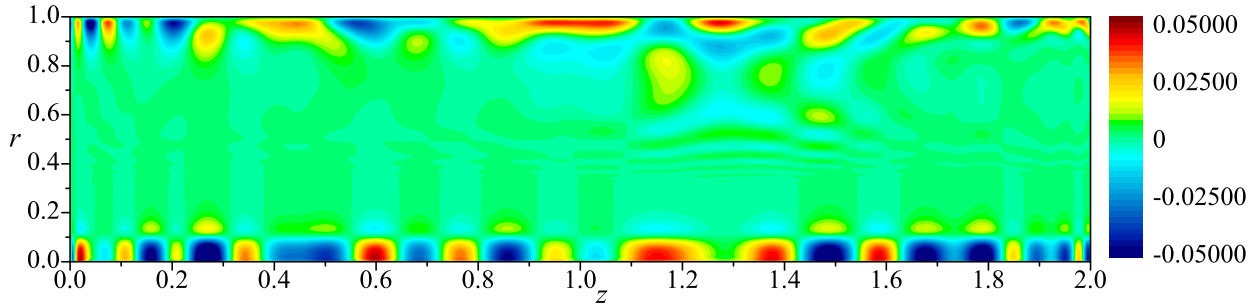


Figure 5.10: Asymmetric eigenvectors for the first undamped frequency, $\omega = 1.6590 + 0.0949i$, with the input parameters $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values.

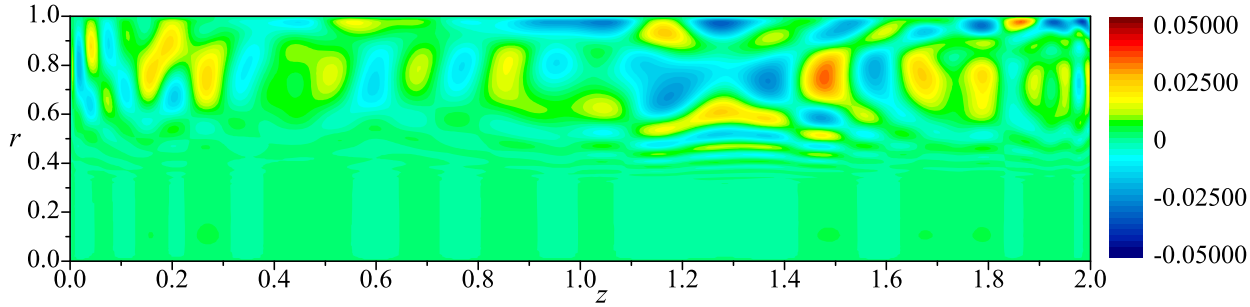
the the r^{-1} singularity at the centerline. Although the formulation is rigorous, their results remain inconclusive given that they were unable to reproduce the results of Abu-Irshaid, Majdalani, and Casalis[119]. It is possible this could be the source of their discrepancy. A final possibility is the presence and effect of the core boundary layer discussed by Majdalani [54] generating vorticity and transferring the largest oscillations to the centerline. This is pure speculation in that this disparity is not observed in all cases. In fact, some input parameters produce uniformly coherent wave forms reminiscent of the vortico-acoustic wave throughout the domain. Regardless of the cause, the solutions predicted by the code are accurate to no less than 10^{-8} for all examples (see Table 5.4).



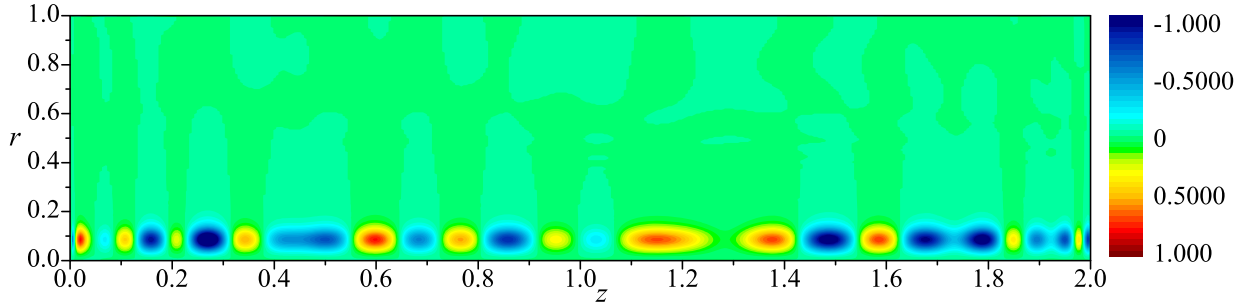
a) LNP asymmetric radial velocity wave for the complex-lamellar model



b) LNP asymmetric tangential velocity wave for the complex-lamellar model



c) LNP asymmetric axial velocity wave for the complex-lamellar model



d) LNP asymmetric pressure wave for the complex-lamellar model

Figure 5.11: Asymmetric contour plots, with $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$.

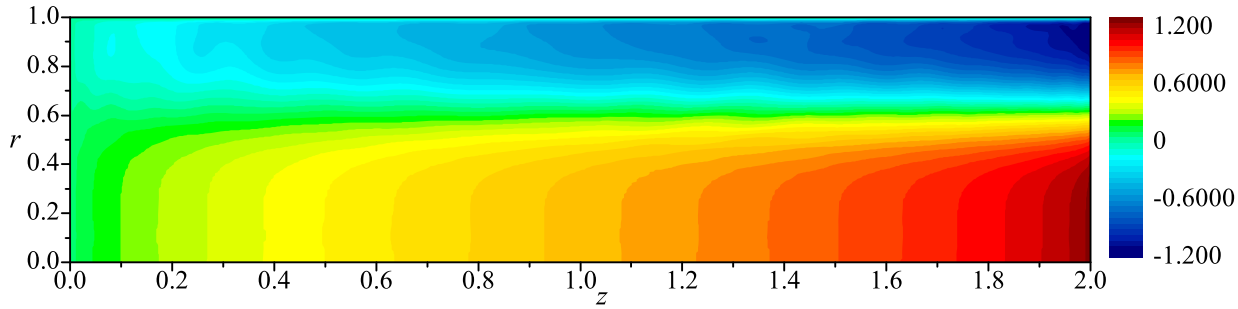
The pressure shift for the axisymmetric example is also interesting. Although it is fairly uniform throughout the domain, this pressure shift in the perturbation will increase the total chamber pressure significantly. Experimentation would be necessary to verify this result.

Figures 5.9–5.11 show the radial, tangential, axial, and pressure waveforms as calculated by the LNP analysis. These plots are generated by determining the first amplified eigenvalue and eigenvector at incremental axial locations along the chamber length. The instantaneous velocity can be constructed in three dimensions by superimposing the perturbation over the base flow. The axisymmetric solutions in Fig. 5.9 suggest a more apparent and cohesive axial oscillation with minimal oscillation in the radial direction. This is especially apparent in the pressure contour where the plot shows oscillations reminiscent of an acoustic wave. Conversely, the results in Fig. 5.11 are much less axially coherent. They embody a great deal of oscillation in both the axial and radial directions. For the radial velocity and pressure profile, the largest oscillations occur near the centerline. The tangential velocity also does, but demonstrates a similar oscillation near the sidewall as well. The axial velocity shows the highest amplitude oscillation near the sidewall with a diminishing amplitude as it travels toward the centerline. The oscillation amplitude is nearly zero from $0 \leq r \leq 0.4$ for the axial fluctuation. This is the expected result of a "parietal" instability.

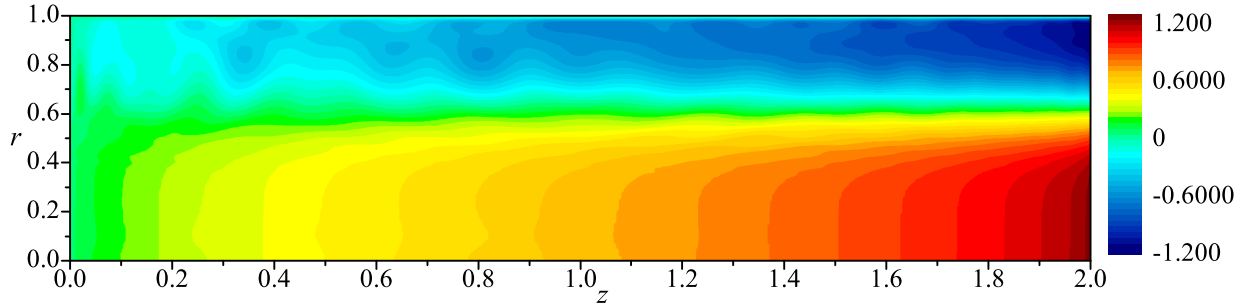
The contour plots in Fig. 5.12 show the spatial degradation as a function of time of the axial velocities along the length of the chamber. Table 5.1 shows the first amplified eigenvalues for the given input parameters at each point along the streamwise direction.

Since the spectrum is a function of axial position, it is important to track how the eigenvalue shifts along the chamber. We see that the greatest effect of hydrodynamic breakdown is near the headwall rather than the endwall as most spatial results would suggest. Considering both the radial and axial solutions together we identify the formation of coherent vortex structures along the sidewall as time increases. The original work by Abu-Irshaid, Majdalani, and Casalis [119] also concluded the region most susceptible to hydrodynamic breakdown was near the headwall. This unique result was scrutinized heavily

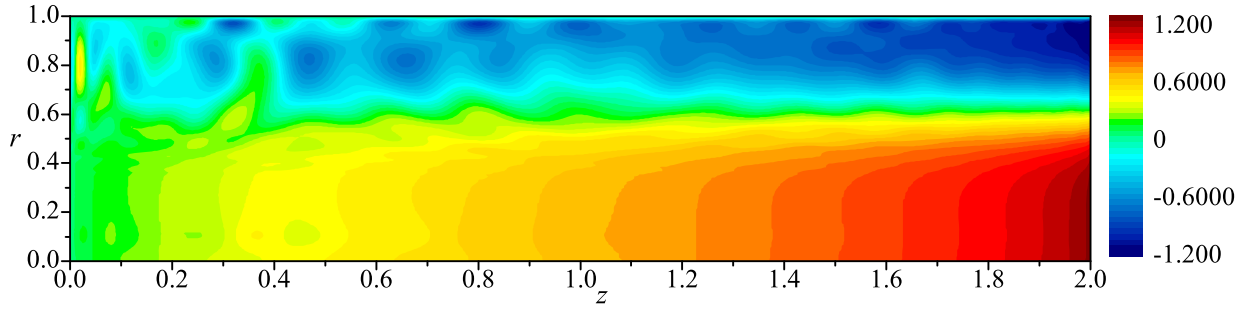
but remained under question. This independent study verifies that this is indeed a probable occurrence according to an LNP approach by showing that the magnitude of the instability wave initially exceeds that of the base flow in the headwall region even before amplification occurs. Traditionally, iso-n plots are used to illustrate the parring of temporal and spatial instability. However, this lends little visual insight to the effect of the hydrodynamic wave on the instantaneous velocity. Contour plots of the instantaneous velocity directly show the location of highest turbulent breakdown where iso-n plots only identify the location of highest amplification. These two locations may not coincide if the magnitude of the base flow is also varying axially. Unfortunately, this observation should still be handled with care. Although this result seems plausible given that the swirling intensity is highest along the headwall and that the flow undergoes its largest turning of the outer vortex into the inner, this region of highest instability also coincides with the region where the violation of the parallel flow assumption is most severe. This question is expected to be rectified with a bigobal analysis.



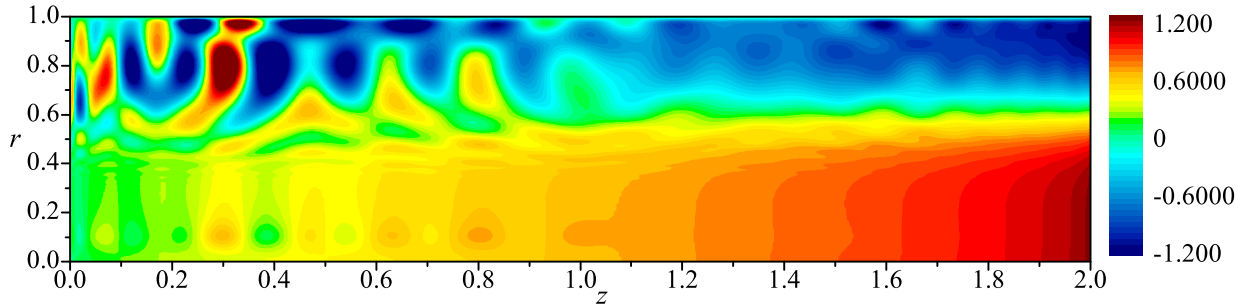
a) Axial velocity contour plot at 1 second



b) Axial velocity contour plot at 3 seconds



c) Axial velocity contour plot at 5 seconds



d) Axial velocity contour plot at 7 seconds

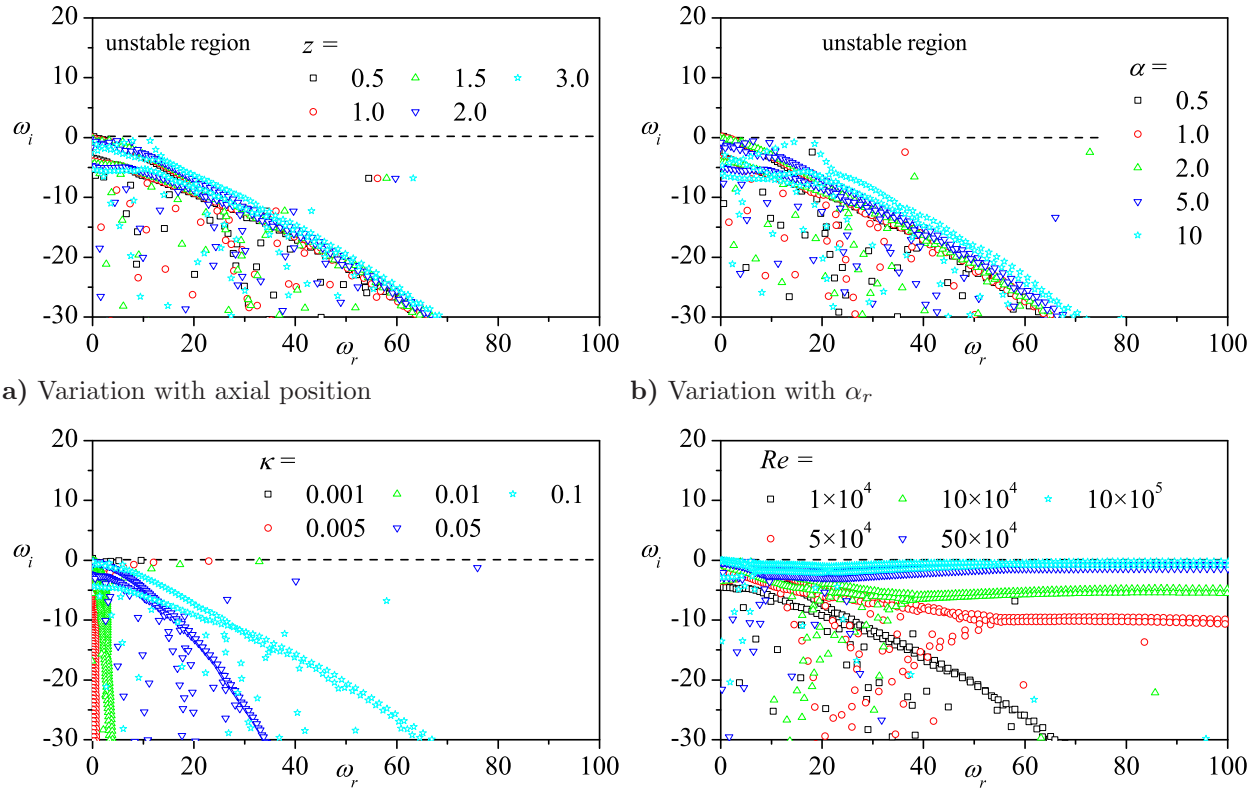
Figure 5.12: Temporal evolution of the first unstable eigenvalue on the axial velocity for the complex-lamellar bidirectional vortex with $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

Table 5.1: The first amplified eigenvalues for $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$.

$z =$	$\omega =$
0.1	$0.68184600813309 + 0.613700646088293i$
0.2	$0.49132772166657 + 0.587229586013676i$
0.3	$0.29487685458900 + 0.570099521022216i$
0.4	$0.09578018608681 + 0.559489701812441i$
0.5	$1.22198008348298 + 0.730967782541060i$
0.6	$1.08339271871411 + 0.681517449717813i$
0.7	$0.94159288566221 + 0.631001125827657i$
0.8	$0.79859780415616 + 0.579679469005072i$
0.9	$0.65647739945641 + 0.527005339381296i$
1.0	$0.51798779045094 + 0.471652699748546i$
1.1	$0.53377639295402 + 0.110796218049251i$
1.2	$0.28955595706607 + 0.139908622010756i$
1.3	$0.02395499346183 + 0.110971728260542i$
1.4	$0.09404299639000 + 0.360686717212237i$
1.5	$1.65899741715896 + 0.094888718198153i$
1.6	$1.53796612918536 + 0.081636458828365i$
1.7	$1.41240149159865 + 0.070012945628646i$
1.8	$1.28295889106999 + 0.059504786156763i$
1.9	$1.15019282233233 + 0.049774919253337i$
2.0	$1.01456100022404 + 0.040599345159635i$

5.4 The Linear Beltramian Bidirectional Vortex

Next we apply the same parametric analysis to the linear Beltramian vortex model. This model differs greatly from the complex lamellar solution since it takes the form of Bessel rather than trigonometric functions. This results in a natural linear decay of the magnitude of axial velocity as the solution travels away from the centerline. Furthermore, the tangential velocity retains axial dependence: a characteristic not previously shown in the complex lamellar model. This weak dependence significantly changes the vorticity profile in the chamber, thus influencing hydrodynamic flow breakdown.



a) Variation with axial position

b) Variation with α_r

c) Variation with κ

d) Variation with Reynolds number

Figure 5.13: Axisymmetric parametric study for several input parameters. Here, $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

5.4.1 Axisymmetric Spectrum

Figure 5.13 refers the same parametric study applied to the complex lamellar profile. Coherent pseudo-spectral lines are identifiable in all four subfigures. We see an interesting nodule protruding from the pseudo-continuous spectral line near $\omega_r = 15$ for the $\alpha = 10$ case in Fig. 5.13b. Although it is a unique spectral structure, it terminates below the stability boundary. The spacing of eigenvalues in the nodule give it the appearance of being "clipped" before generating unstable eigenvalues. A smaller, but similar feature is seen for $z = 3$ in Fig. 5.13a. The effect of κ and Re are evident in Figs. 5.13c–5.13d and parallel those seen in the complex-lamellar solution.

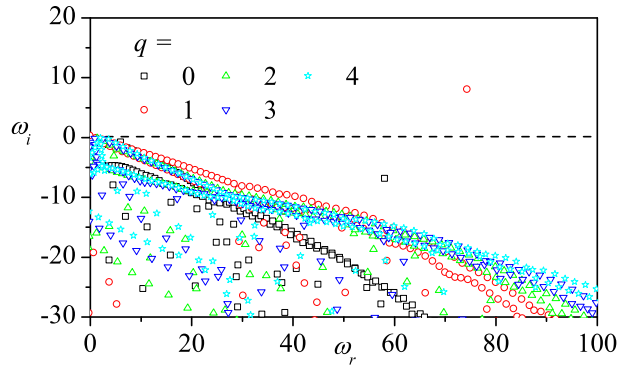


Figure 5.14: The effect of higher tangential mode numbers with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

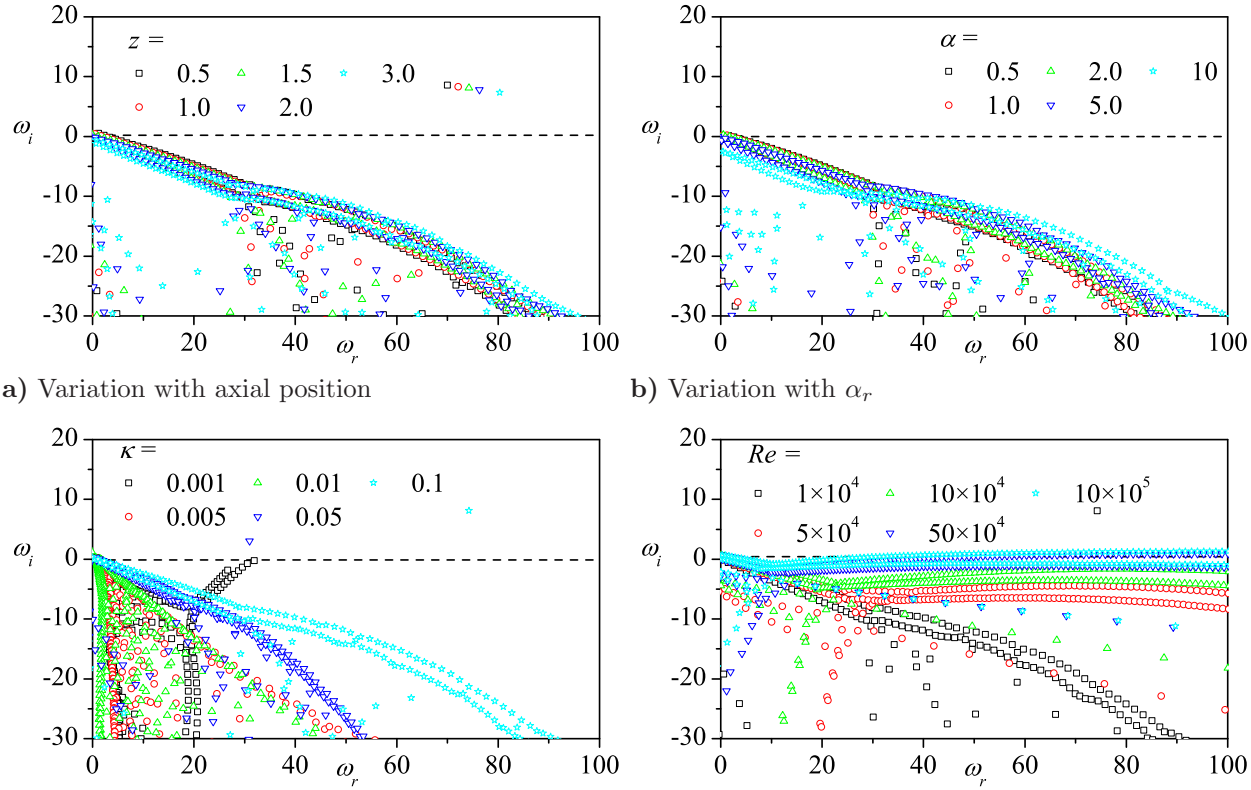
5.4.2 Asymmetric Spectrum

It appears that increasing the tangential mode number past one has a small effect on the overall spectrum. For low frequency eigenvalues, the disparity is very small for $q \geq 1$. This increases slightly at higher frequency but does not excite more amplified eigenmodes.

The first asymmetric eigenmode drastically changes the shape of the pseudo-continuous spectral lines in Fig. 5.15. As we have seen before, it flattens out the spectrum while exciting low frequency eigenvalues in the unstable region. The variation with α is once again surprising. The asymmetric mode forces strong coherence back into the spectrum and actually reduces the number of amplified eigenvalues. The other figures are in agreement with those for the complex lamellar solutions such that κ and Re have comparable spectra.

5.4.3 Multiple Mantles

Figure 5.16 shows a significant breaking of the coherent spectral line for increased mantles. The vertical and lateral distance between eigenvalues is increased and the coupled spectral lines are spread further apart. We also see an increase in the number of unstable modes for higher mantle numbers. From a physical perspective, it is likely that multidirectional flow



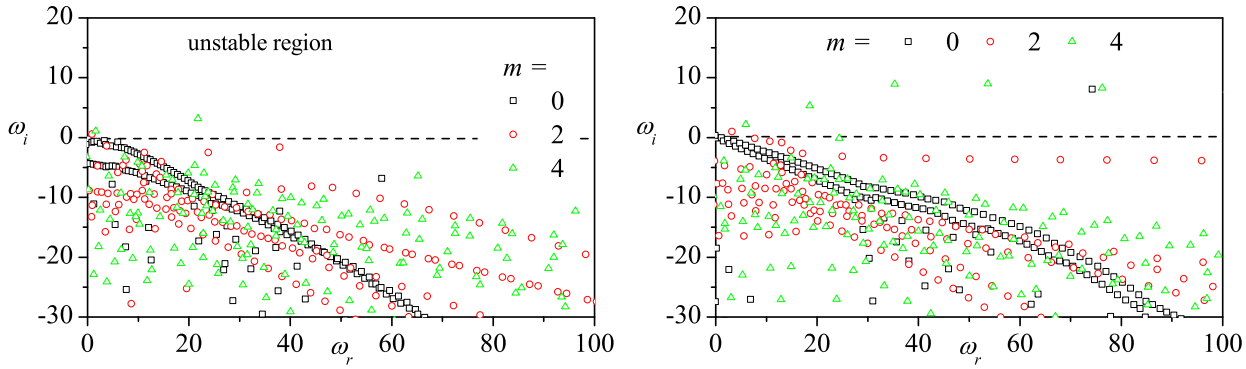
a) Variation with axial position

b) Variation with α_r

c) Variation with κ

d) Variation with Reynolds number

Figure 5.15: Asymmetric parametric study for several input parameters. Here $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.



a) Axisymmetric perturbations ($q = 0$) **b)** Asymmetric perturbations ($q = 1$)

Figure 5.16: The effect of multiple mantles on the temporal spectrum for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

will be more susceptible to hydrodynamic breakdown due to the increased number of axial shear layers formed by the concentric vortex tubes. If this argument is plausible, the number of possible mantles in a sustained vortex chamber may be directly limit by its propensity toward hydrodynamic breakdown.

5.4.4 Amplified Frequencies

The axisymmetric wave forms in Fig. 5.17 are reminiscent to those for the comparable case with the complex lamellar base flow. We see slight oscillations near the centerline as well as the sidewall. The exact causes of the oscillations at the centerline are not fully known; however the computed eigenvectors solve the governing equations with remarkable accuracy. Table 5.4 verifies this statement. We do find the expected wave form in Fig. 5.19. The axial profile seems to represent a parietal instability wave in which large amplitude oscillations occur near the surface at $r = 1$. This behavior occurs independently of the radial, tangential, and pressure oscillations that reach their largest amplitudes near the centerline.

In this case, the axisymmetric contour plots do not show the same acoustic wave character as seen with the complex-lamellar base flow. Rather, the results in Fig. 5.18 show little-to-no oscillation throughout the majority of the chamber. In general, if oscillations

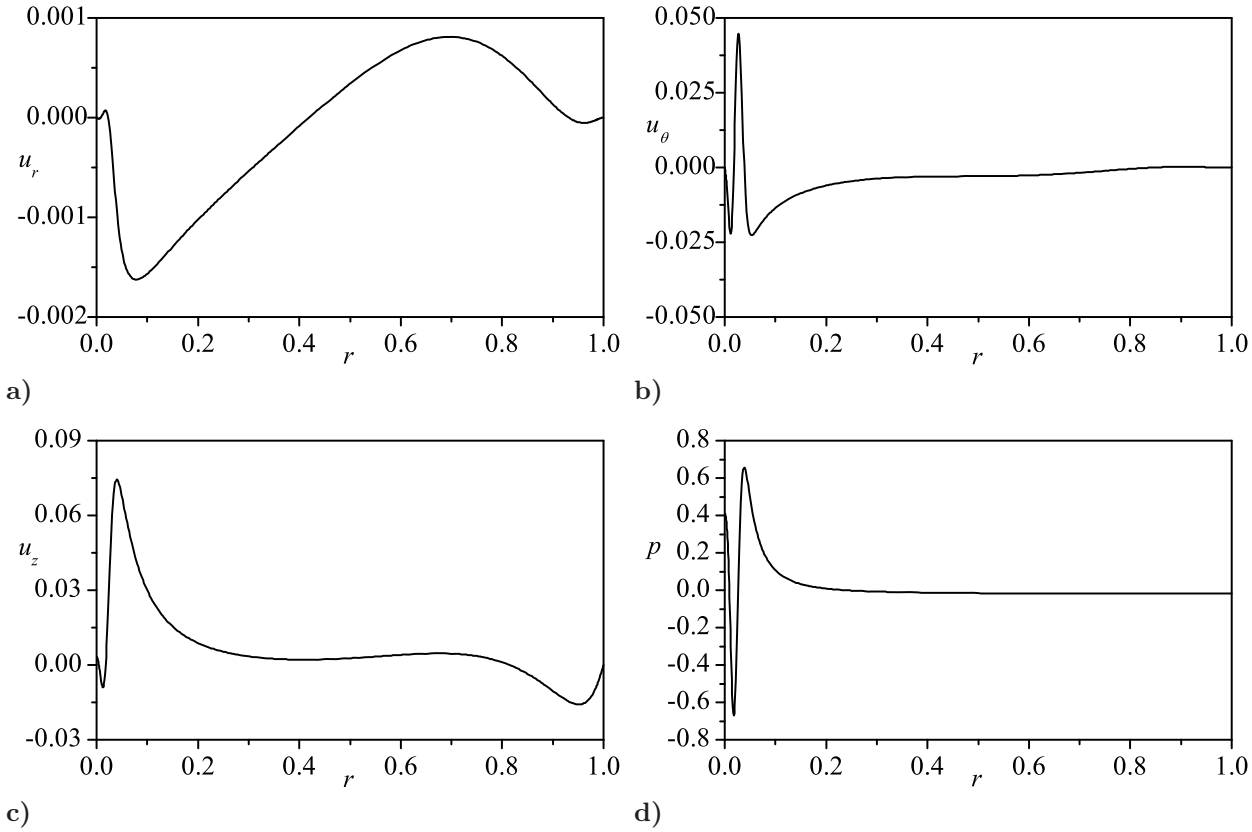
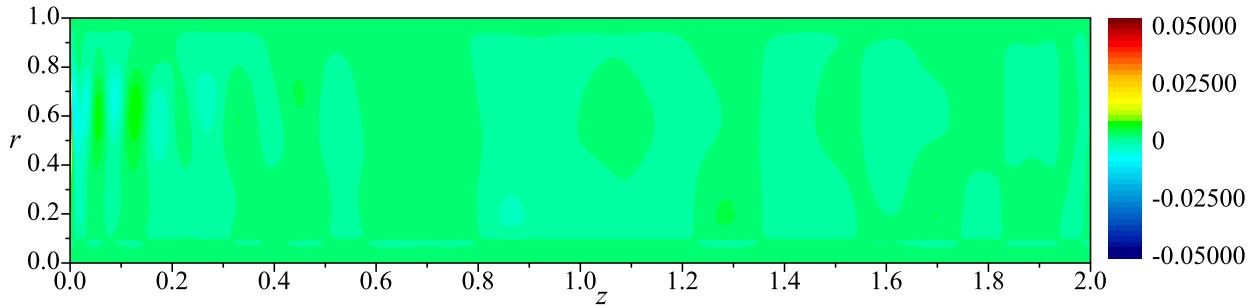
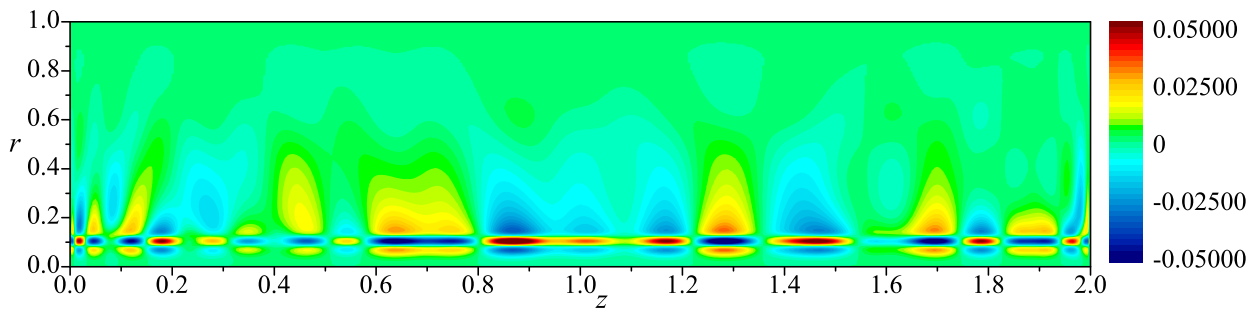


Figure 5.17: Axisymmetric eigenvectors for the first undamped frequency, $\omega = 0.5645 + 0.2122i$, with the input parameters $q = 0$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values.

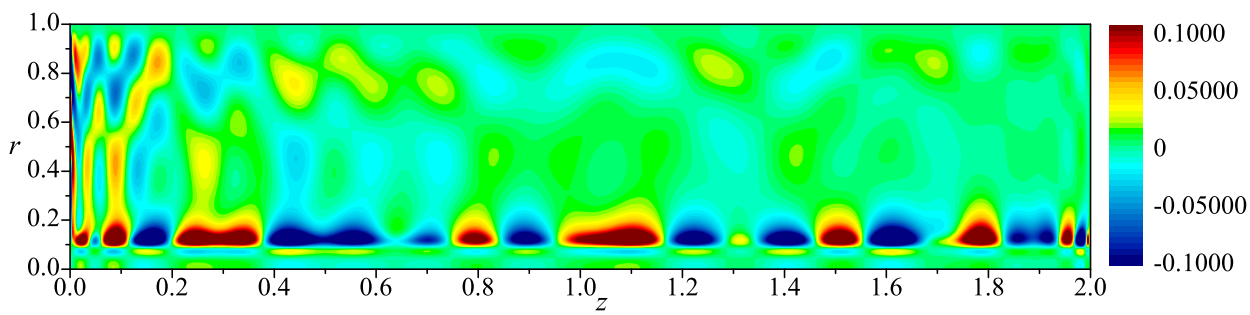
occur, they are confined to a region near the centerline. This region corresponds reasonably well with the forced core region of the tangential velocity. Figure 5.20 illustrates the asymmetric solution. Again, oscillations are confined to a similar centerline region in the radial and pressure plots. The tangential velocity shares similar oscillations near the centerline to those of the pressure, however, it also shows radial oscillations mimicking the axial waveform deeper in the chamber. Once again, the axial velocity shows the classic parietal instability waveform with the largest oscillations near the sidewall and diminishing amplitude toward the centerline. Near the headwall, the waveform appears to be stretched from the headwall toward the endwall. This shape suggests the effect of the core vortex



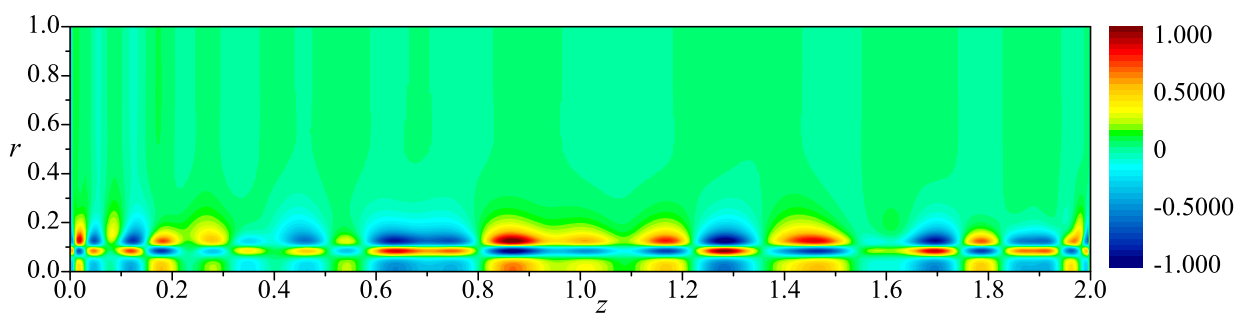
a) LNP axisymmetric radial velocity wave for the linear Beltramian model



b) LNP axisymmetric tangential velocity wave for the linear Beltramian model



c) LNP axisymmetric axial velocity wave for the linear Beltramian model



d) LNP axisymmetric pressure wave for the linear Beltramian model

Figure 5.18: Axisymmetric contour plots, with $q = 0$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$.

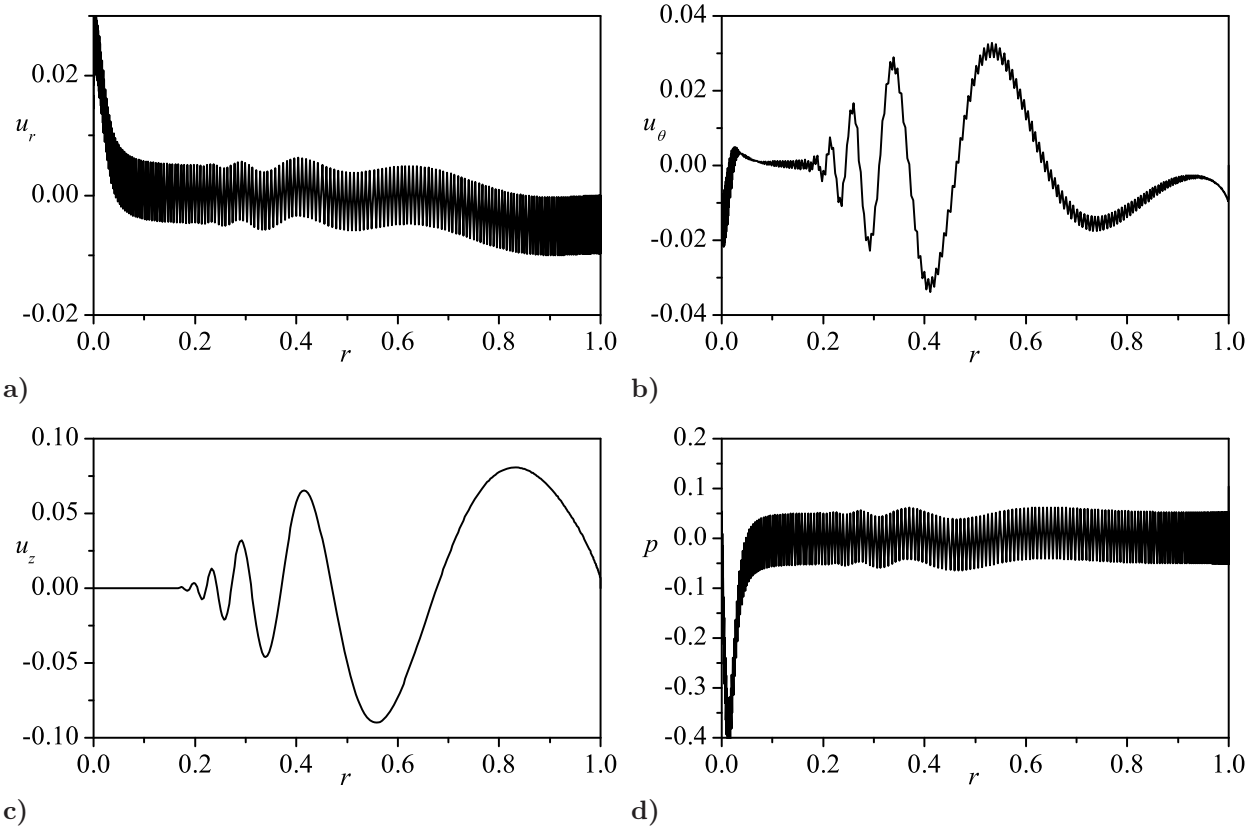
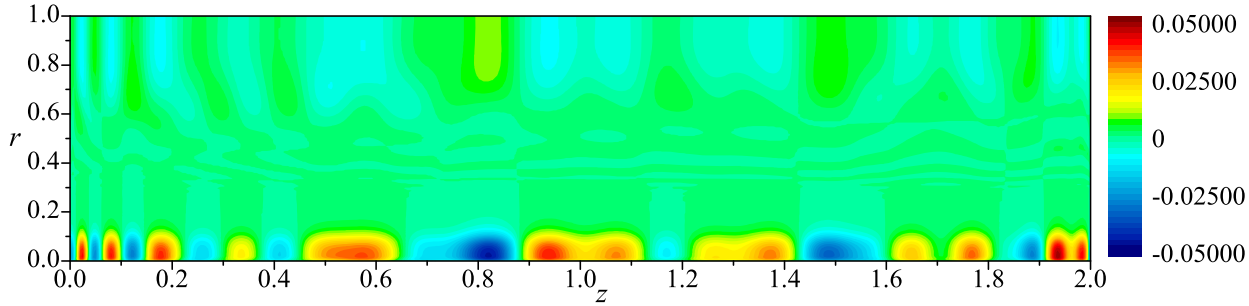


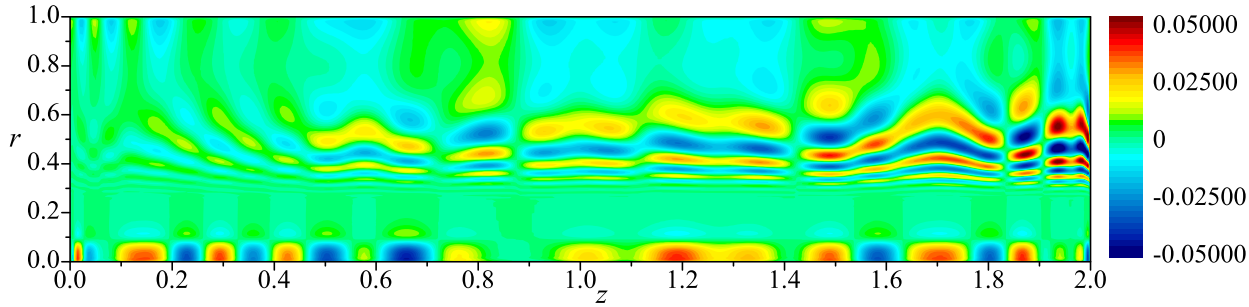
Figure 5.19: Asymmetric eigenvectors for the first undamped frequency, $\omega = 0.6931 + 0.1220i$, with the input parameters $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$. See Table 5.4 for error values.

pulling the outer flow inwardly at the headwall. The oscillations appear to be pulled inward as well.

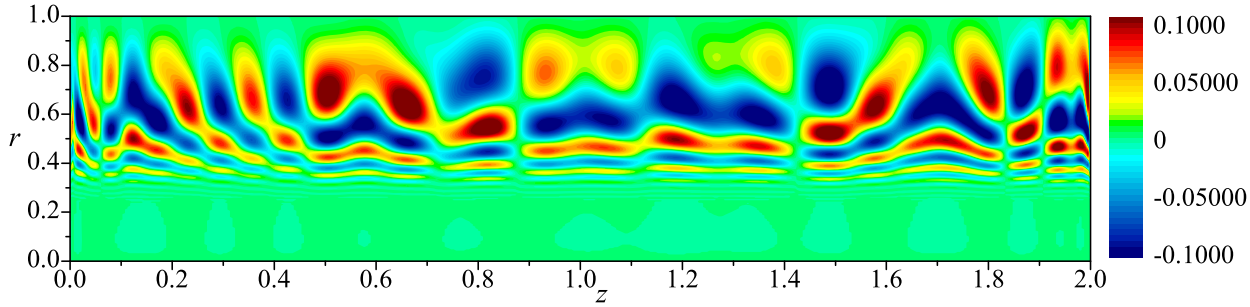
Contour plots along the length of the chamber and at discrete cross-sections are shown in Fig. 5.21. As usual we consider only the first amplified eigenvalue explicitly given in Table 5.2. The higher amplitude base flow is evident in color scale. It can also be inferred that the amplitudes of these eigenmodes are slightly larger than those of the complex-lamellar in order to cause such large diversion from the stable streamlines. The density of the contour lines in certain regions gives further visual perception of the spatial propagation of the hydrodynamic wave.



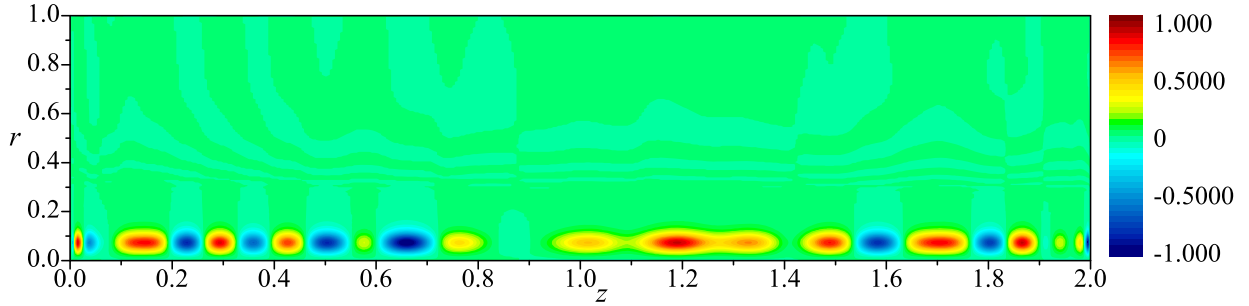
a) LNP asymmetric radial velocity wave for the linear Beltramian model



b) LNP asymmetric tangential velocity wave for the linear Beltramian model

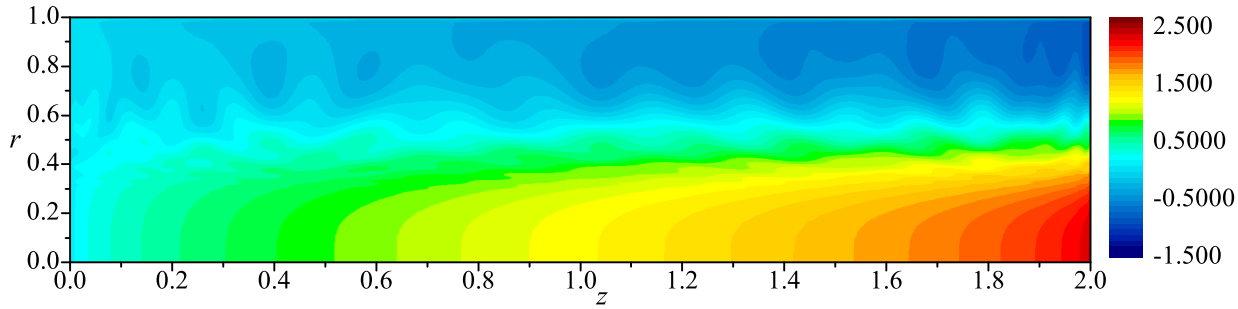


c) LNP asymmetric axial velocity wave for the linear Beltramian model

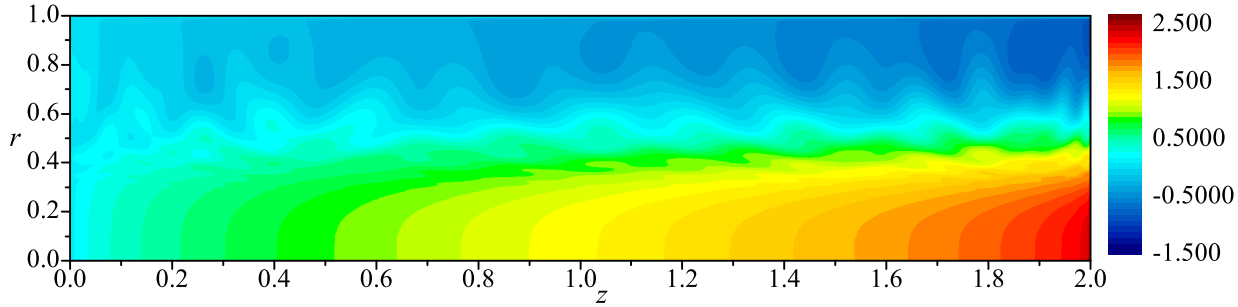


d) LNP asymmetric pressure wave for the linear Beltramian model

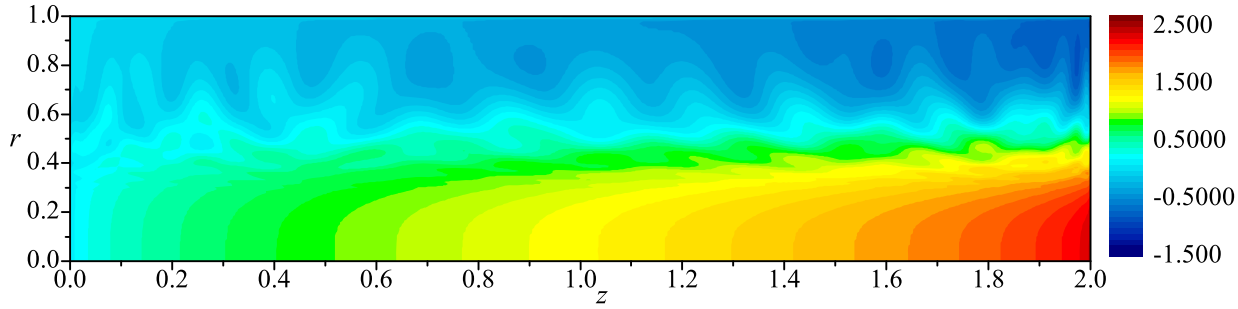
Figure 5.20: Asymmetric contour plots, with $q = 1$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$.



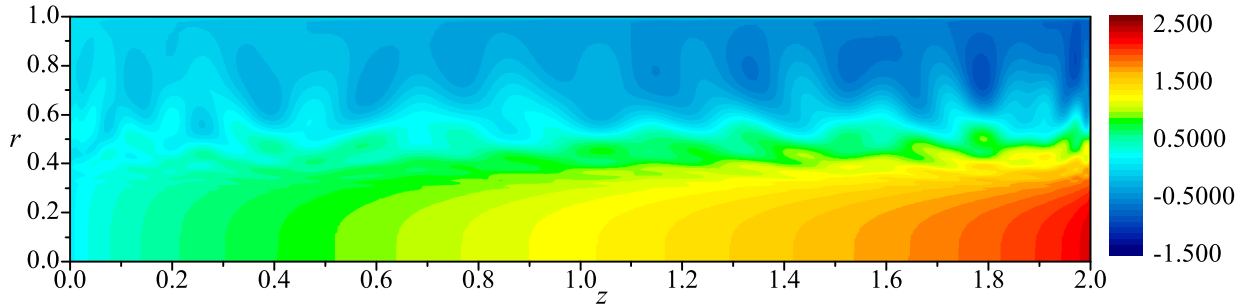
a) Axial velocity contour plot at 1 second



b) Axial velocity contour plot at 3 seconds



c) Axial velocity contour plot at 5 seconds



d) Axial velocity contour plot at 7 seconds

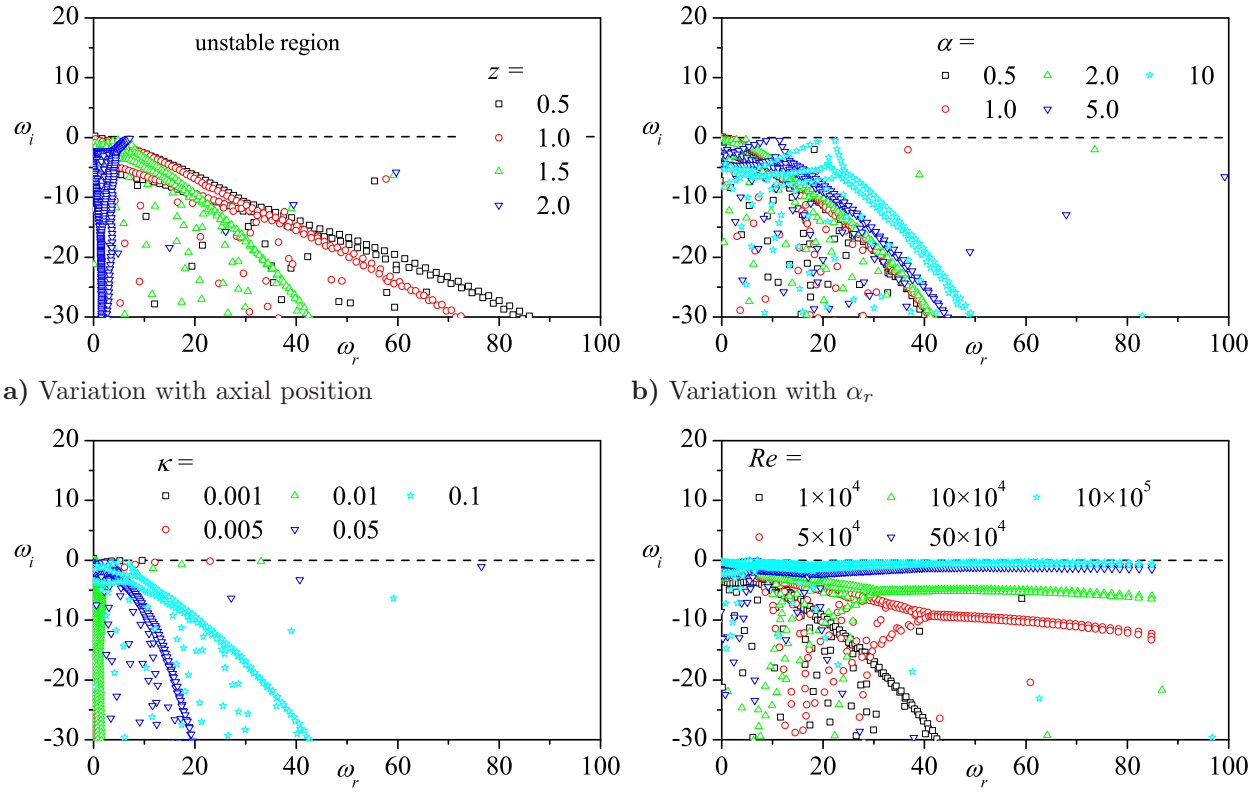
Figure 5.21: Temporal evolution of the first unstable eigenvalue on the axial velocity for the linear Beltramian bidirectional vortex with $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

Table 5.2: The first amplified eigenvalues for $q = 1$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$.

$z =$	$\omega =$
0.1	$1.003524516285300 + 0.063006699912048i$
0.2	$0.983203303998358 + 0.070525142990520i$
0.3	$0.962413560310940 + 0.077126577015609i$
0.4	$0.941239680763080 + 0.082991388473565i$
0.5	$0.919740943741555 + 0.088252263574092i$
0.6	$0.897960793922591 + 0.093009439643143i$
0.7	$0.875932257905165 + 0.097340366348927i$
0.8	$0.853681271789386 + 0.101306061549143i$
0.9	$0.831228811491858 + 0.104955425919258i$
1.0	$0.808592303430979 + 0.108328255271818i$
1.1	$0.785786584189617 + 0.111457390382350i$
1.2	$0.762824567804711 + 0.114370284356712i$
1.3	$0.739717716521766 + 0.117090163019887i$
1.4	$0.716476379315972 + 0.119636896906019i$
1.5	$0.693110037010008 + 0.122027662938465i$
1.6	$0.669627482111685 + 0.124277451587660i$
1.7	$0.646036951377310 + 0.126399457087283i$
1.8	$0.622346225043022 + 0.128405380423409i$
1.9	$0.598562698236634 + 0.130305662331893i$
2.0	$0.574693436312902 + 0.132109666991328i$

5.5 The Harmonic Beltramian Bidirectional Vortex

Lastly, we present a similar parametric analysis of the harmonic Beltramian vortex model. There are strong similarities between this model and its linear counterpart. Its unique characteristics include the axial dependence of its radial velocity profile as well as the sinusoidal axial dependence of its axial velocity. This sinusoidal dependence corresponds to a half sine extending between $0 \leq z \leq l$. This makes direct comparison for long chambers slightly more challenging in that both l and z must be accounted for.



a) Variation with axial position

b) Variation with α_r

c) Variation with κ

d) Variation with Reynolds number

Figure 5.22: Axisymmetric parametric study for several input parameters. Here, $q = 0$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$ unless varied on the graph.

5.5.1 Axisymmetric Spectrum

Again we see an abrupt clipping of the spectrum at the $\omega_i = 0$ line. Almost no amplified eigenvalues are predicted for the axisymmetric spectrum under any of the parameter combinations considered (see Fig. 5.22).

5.5.2 Asymmetric Spectrum

The axisymmetric study suggests this flow is stable for the range of low frequency eigenvalues shown in Fig. 5.22. Conversely, Figs. 5.23–5.24 show an increase in unstable modes over the range of parameters for the first asymmetric mode, $q = 1$. Again, the $\alpha = 3$ wave number

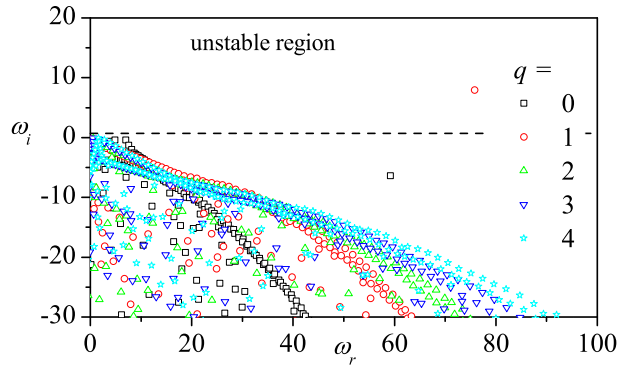


Figure 5.23: The effect of higher tangential mode numbers with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$ unless varied on the graph.

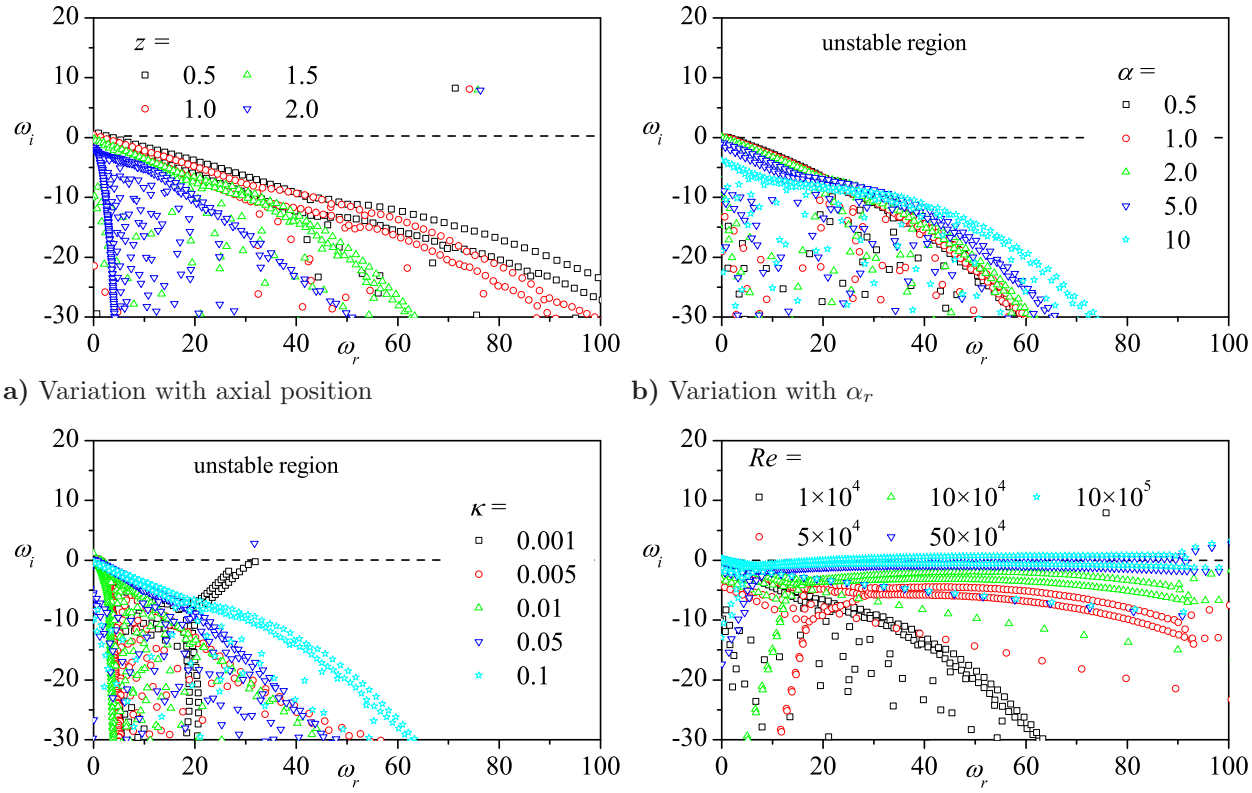
produces a discrete and repeatable unstable mode. For this particular base flow, this mode occurs near $\omega = 70 + 10i$. Beyond this mode, the usual range of amplified eigenvalues appear at low frequency or high Reynolds number.

5.5.3 Multiple Mantles

Figure 5.25 continues to show a breakdown of the coherent spectral line with an increasing number of mantles. The amplified modes appear at nearly the same circular frequencies as they do in the linear Beltramian base flow. They do, however, show a lower rate of amplification due to the smaller complex component of the eigenvalue.

5.5.4 Amplified Frequencies

The axisymmetric modes continue to produce larger oscillations near the centerline than the wall. Figures 5.26–5.28 are comparable to Figs. 5.17–5.19 in amplitude and waveform, respectively. Again, the largest oscillations appear near the centerline while the majority of the chamber remains nearly undisturbed for u_r, u_θ, u_z, and p. This is likely a result of the



a) Variation with axial position

b) Variation with α_r

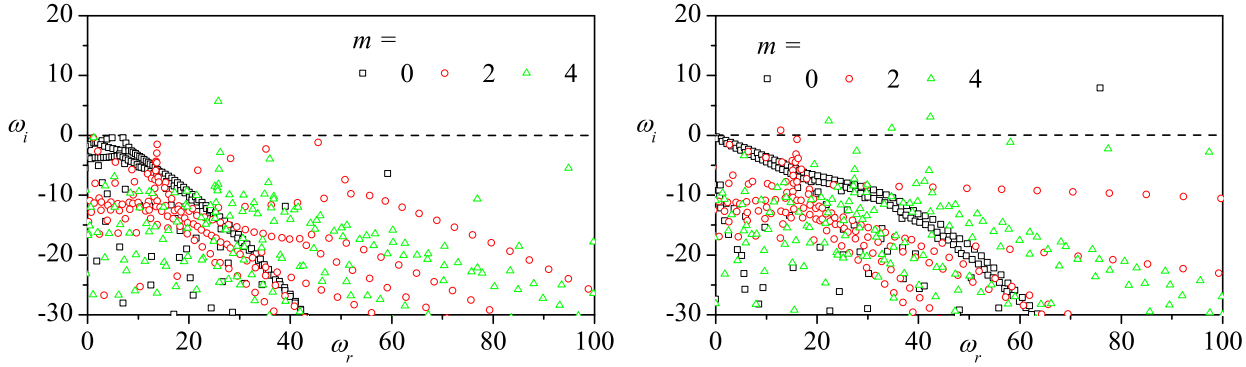
c) Variation with κ

d) Variation with Reynolds number

Figure 5.24: Asymmetric parametric study for several input parameters. Here $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$ unless varied on the graph.

core vortex shear layer of the tangential velocity. The symmetry between the two Beltramian solutions is expected given the close relationship between the two base flows.

The results in Fig. 5.27 show little-to-no oscillation throughout the majority of the chamber for the axisymmetric Beltramian solutions. Once again, confined to a region near the centerline. Figure 5.29 illustrates the asymmetric solution. Again, oscillations are confined to a similar centerline region in the radial and pressure plots. The pressure profile indicates a unique, high amplitude oscillation near the exit plane. This is unique to this solution. The tangential and axial velocities share similar wave forms but higher amplitudes are found in the axial solution. The axial velocity fluctuation is not stretched



a) Axisymmetric perturbations ($q = 0$) **b)** Asymmetric perturbations ($q = 1$)

Figure 5.25: The effect of multiple mantles on the temporal spectrum for $\alpha = 3$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$.

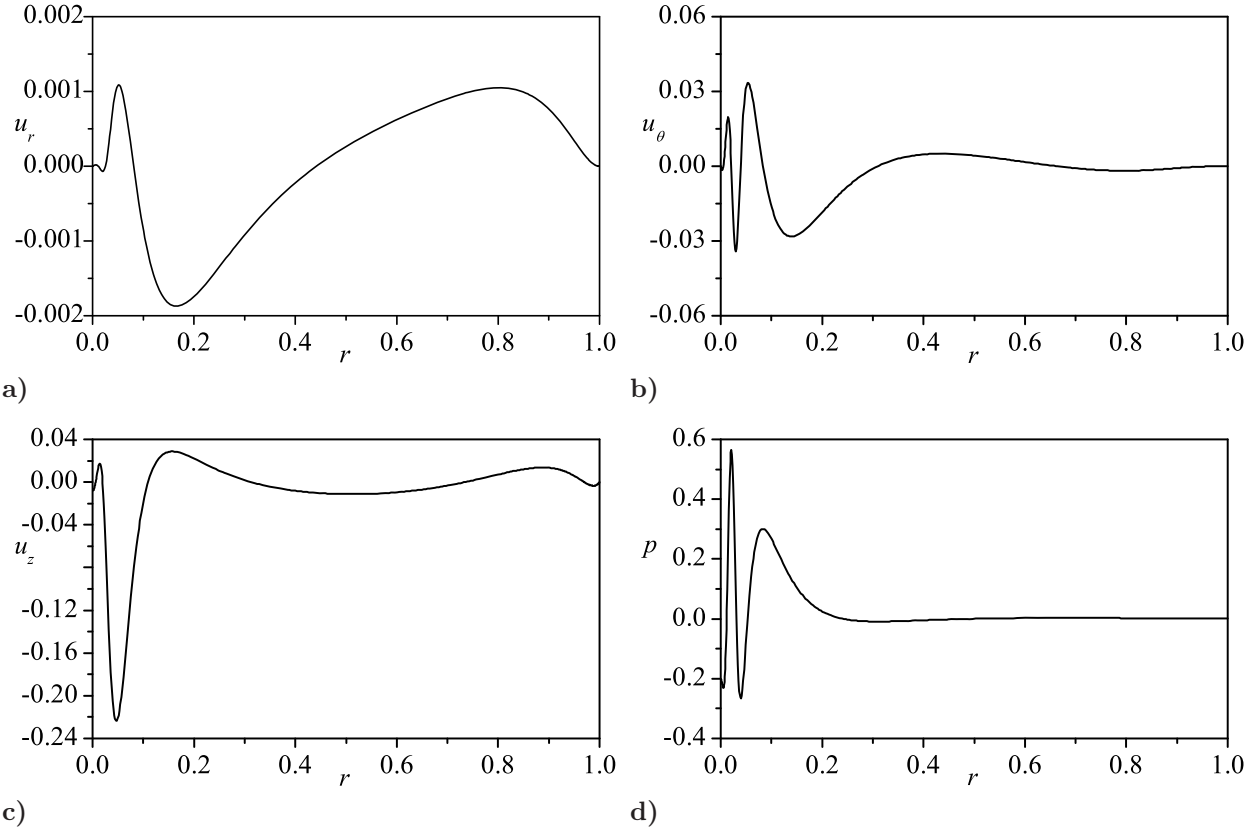
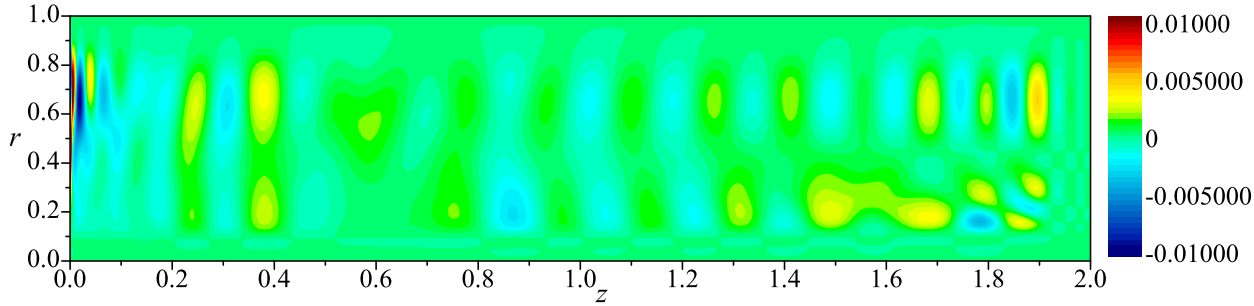
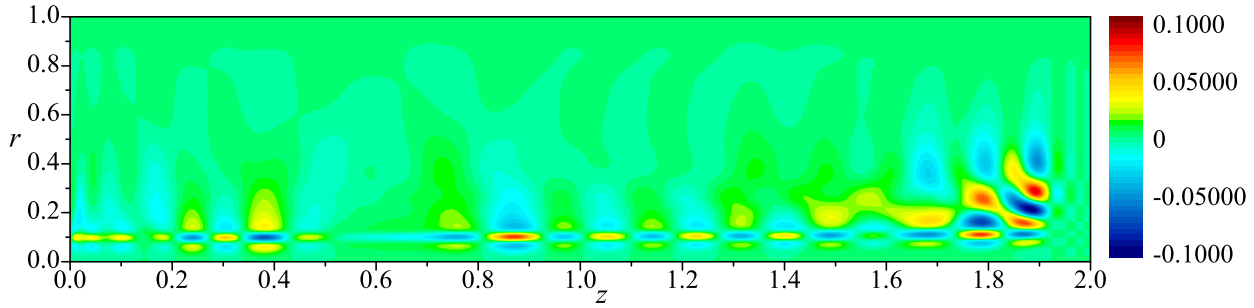


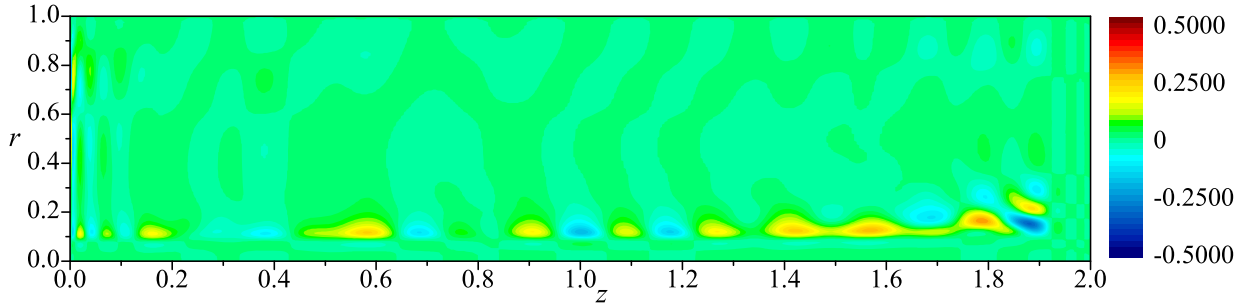
Figure 5.26: Axisymmetric eigenvectors for the first undamped frequency, $\omega = 0.1891 + 0.1000i$, with the input parameters $q = 0$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. See Table 5.4 for error values.



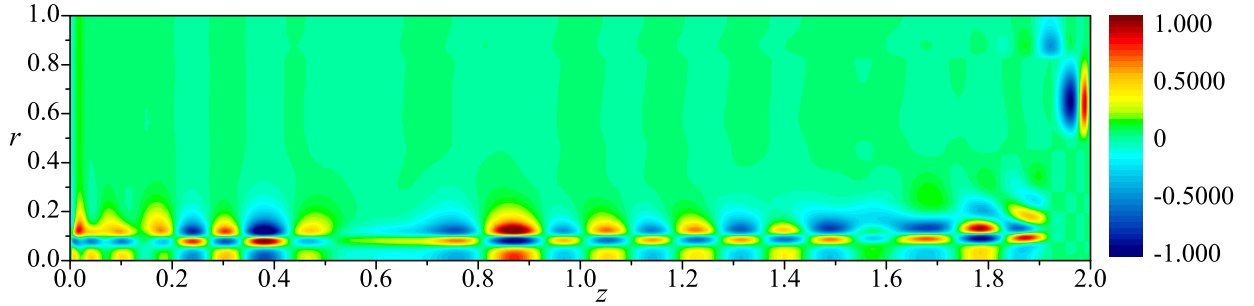
a) LNP axisymmetric radial velocity wave for the harmonic Beltramian model



b) LNP axisymmetric tangential velocity wave for the harmonic Beltramian model



c) LNP axisymmetric axial velocity wave for the harmonic Beltramian model



d) LNP axisymmetric pressure wave for the harmonic Beltramian model

Figure 5.27: Axisymmetric contour plots, with $q = 0$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$.

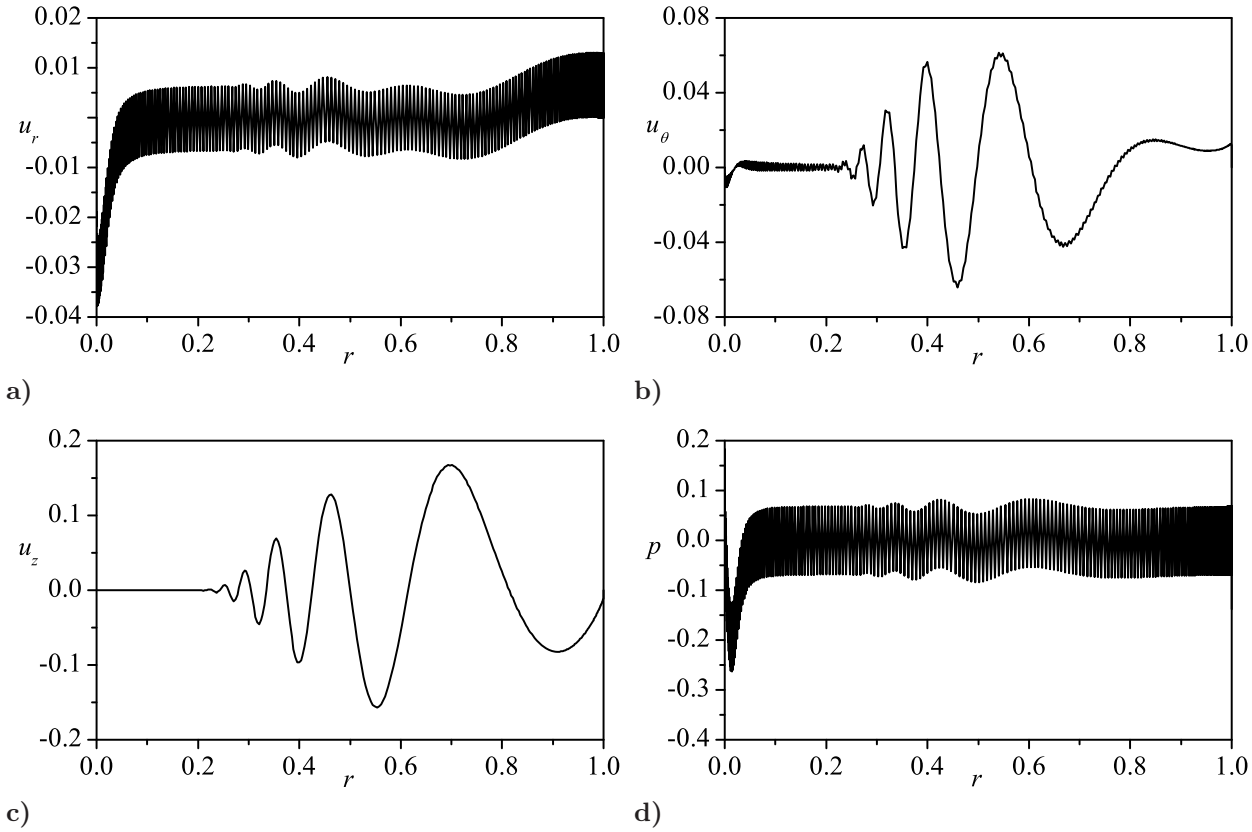
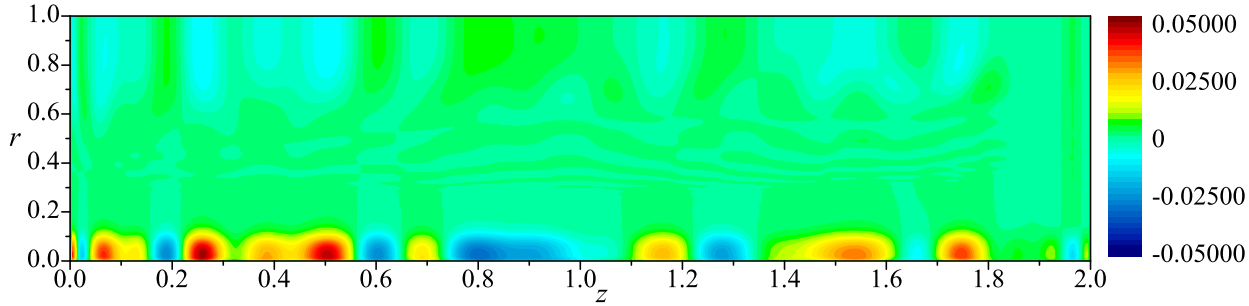


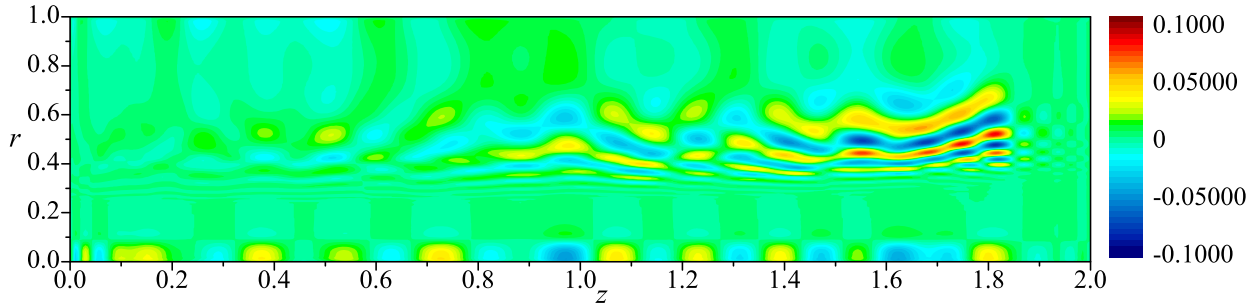
Figure 5.28: Asymmetric eigenvectors for the first undamped frequency, $\omega = 0.6262 + 0.0331i$, with the input parameters $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$. See Table 5.4 for error values.

toward the inner vortex as seen in the other figures even though it bears other similarities to the linear Beltramian solution.

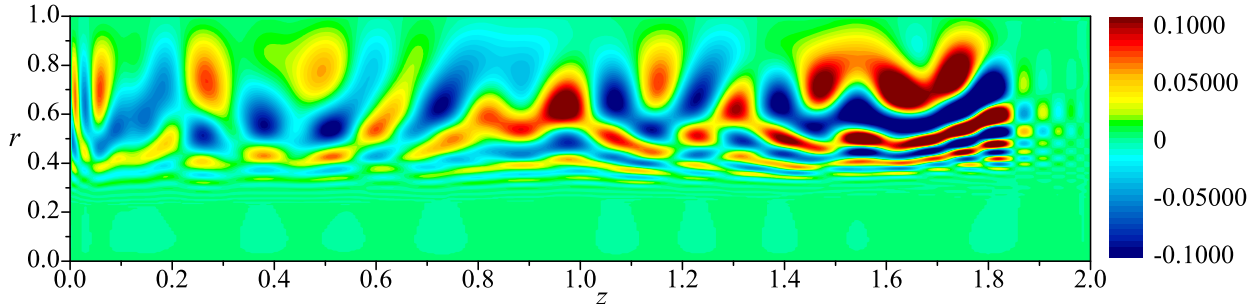
Figure 5.30 shows the temporal development of the hydrodynamic instability along the chamber length. This model contains no low frequency amplified eigenmodes beyond an axial position of $z = 1.6$. Beyond this position, the first amplified eigenvalue jumps outside the range of numerically accurate eigenvalues. Thus, Fig. 5.30 continues to track the first eigenvalue as it becomes stable near the endwall. The eigenvalues and corresponding axial positions are found in Table 5.3.



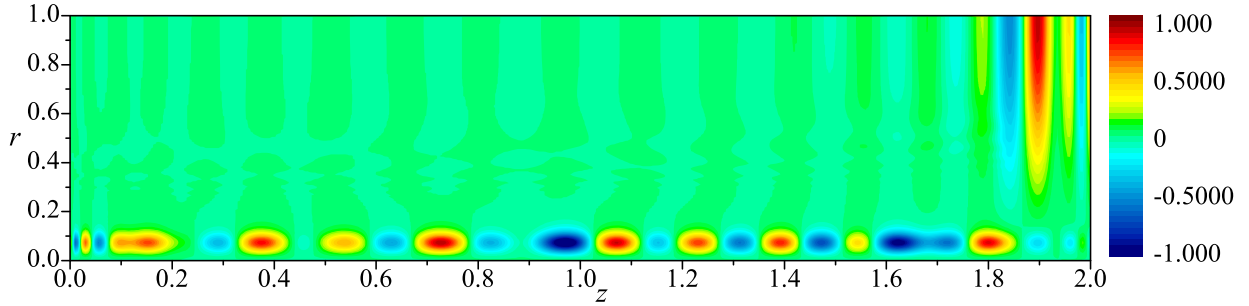
a) LNP asymmetric radial velocity wave for the harmonic Beltramian model



b) LNP asymmetric tangential velocity wave for the harmonic Beltramian model

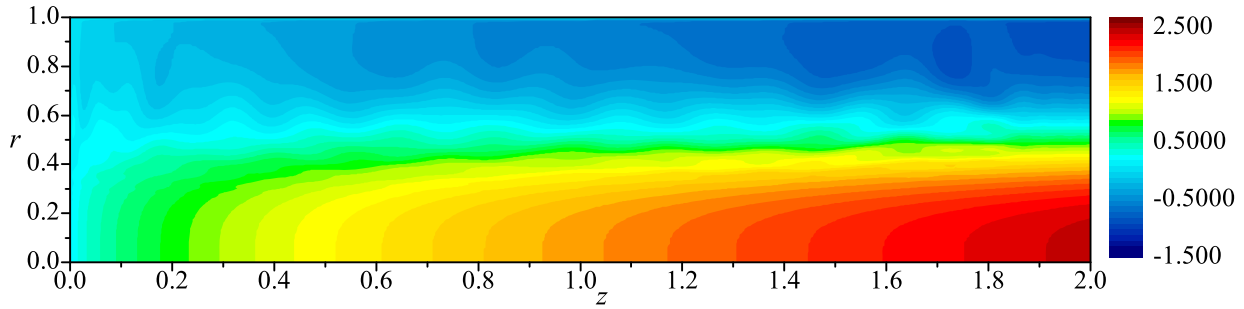


c) LNP asymmetric axial velocity wave for the harmonic Beltramian model

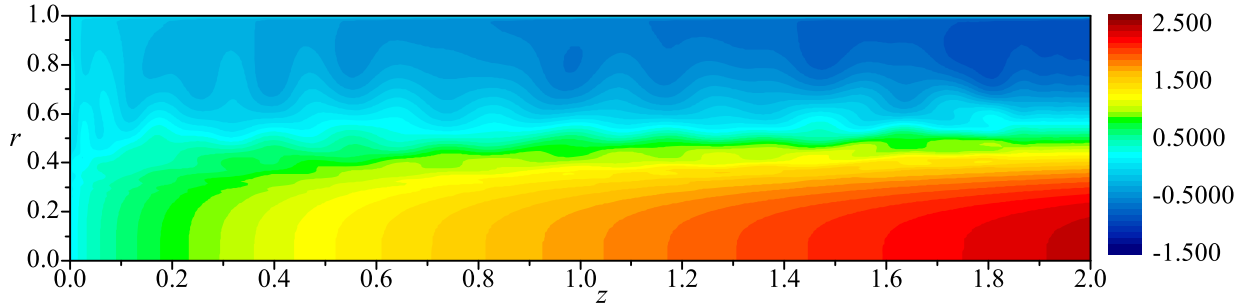


d) LNP asymmetric pressure wave for the harmonic Beltramian model

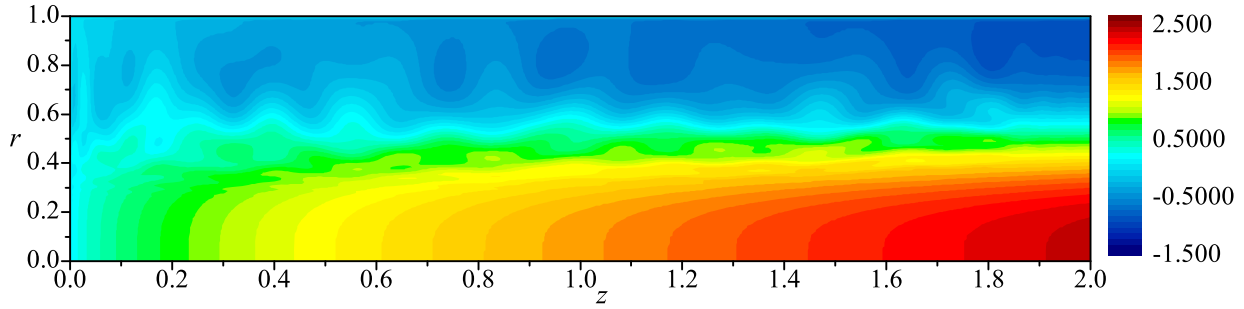
Figure 5.29: Asymmetric contour plots, with $q = 1$, $\alpha = 0.5$, $Re = 10,000$, and $\kappa = 0.1$.



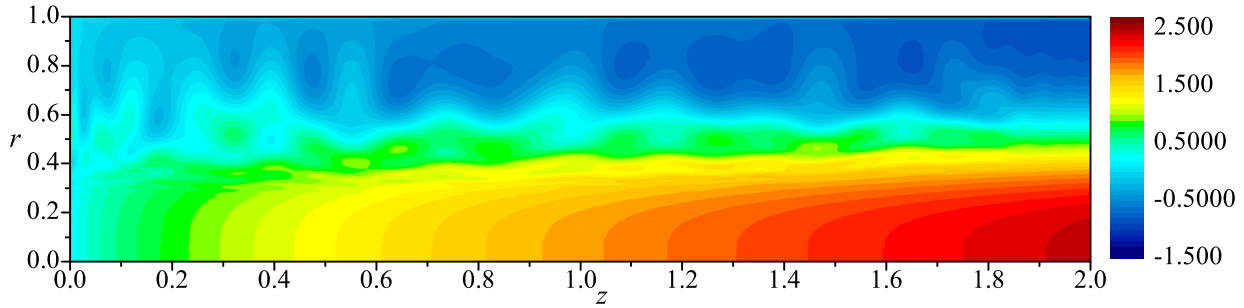
a) Axial velocity contour plot at 1 second



b) Axial velocity contour plot at 3 seconds



c) Axial velocity contour plot at 5 seconds



d) Axial velocity contour plot at 7 seconds

Figure 5.30: Temporal evolution of the first unstable eigenvalue on the axial velocity for the Harmonic Beltramian bidirectional vortex with $q = 1$, $\alpha = 0.5$, $z = 1.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$.

Table 5.3: The first amplified eigenvalues for $q = 1$, $\alpha = 0.5$, $Re = 10,000$, $l = 2$, and $\kappa = 0.1$.

$z =$	$\omega =$
0.1	0.95717385153943 + 0.15285395453130i
0.2	0.92728649501637 + 0.18168539908038i
0.3	0.89608840607973 + 0.20129801990820i
0.4	0.86513351516908 + 0.21311464789768i
0.5	0.83564202225154 + 0.21811653643692i
0.6	0.80836118817958 + 0.21696584066040i
0.7	0.78358341622263 + 0.21008492814540i
0.8	0.76100539070175 + 0.19777343968856i
0.9	0.73947431528243 + 0.18050440910282i
1.0	0.71736420751419 + 0.15937043137887i
1.1	0.69433779406064 + 0.13591150261076i
1.2	0.67219334632560 + 0.11110814697065i
1.3	0.65294274645569 + 0.08529294081106i
1.4	0.63750380597771 + 0.05894978581570i
1.5	0.62618252009516 + 0.03305088024017i
1.6	0.61955726893038 + 0.00901753689037i
1.7	0.61894802707506 - 0.011480920318842i
1.8	0.62582797648452 - 0.026744920874072i
1.9	0.64167029799103 - 0.035074893466655i
2.0	0.66794817270807 - 0.034771266990080i

5.6 Closing Remarks on the LNP Applied to the BV

When considering these results, it is necessary to understand that these parametric studies are a small sample of the possible configurations. Without experimental data and fixed input parameters, real world cases cannot be predicted or verified. The nature of the parametric studies are still valuable in that they give a qualitative description of the hydrodynamic stability properties associated with a given flowfield. Specific examples, such as the wave form plots, are discrete examples. The very nature of turbulent breakdown suggests a very wide range of wave forms at many scales. We can see the variety of wave forms by slightly modifying the input parameters and recovering significantly different results.

Numeric backsubstitution shown in Table 5.4 justifies any unusual results by verifying the accuracy in which the eigenvector solves the equations and boundary conditions. It is difficult (if at all possible) to present an all-encompassing result that accounts for the large disparity in scales and wave forms. The problem of flow breakdown to large vortices and then into successively smaller vortices is simply too complex. Thus, we should focus on the most amplified and lowest frequency (lowest energy) eigenvalues and eigenmodes. From these results we can present tools that suggest the most problematic design parameters for real applications.

Little discussion is made regarding the tangential velocity perturbations. Even though this is a swirl dominated base flow, this fluctuation is minuscule compared to the base flow for small time. Its inclusion is merely for completeness. The inclusion of the tangential component of the base flow, however, holds significant influence over the spectral results and cannot be neglected. When compared to the LNP results for the Taylor-Culick flow, the presence of a tangential mean flow velocity reduces the number of amplified eigenvalues. This idea of centrifugal flow promoting stable behavior as suggested by Eloy and Le Dizès [10] is corroborated.

This chapter is overshadowed by the forthcoming biglobal analysis; however, it confirms the results obtained in the original hydrodynamic study by Abu-Irshaid, Majdalani, and Casalis [119]. This preliminary investigation suggests that parietal vortex shedding is indicative of asymmetric hydrodynamic instability. It also suggests that axisymmetric modes tend to be more stable than higher modes though all three mean flow models appear to be inherently stable. At first, the presence of amplified eigenvalues existing in many of the case studies presented in this chapter appear to be in contradiction with this statement. However, each amplified eigenvalue is accompanied by one or more damped eigenvalues at or very near the same circular frequency. In all cases the average growth rate near a given circular frequency is negative; thus implying overall stability. While it is possible to excite a specific amplified frequency, any deviation will return the flow to a stable state. If the

flow is driven at an undamped frequency, Fig. 5.12 indicated a quicker turbulization of the complex-lamellar flow than either Beltramian solution (see Fig. 5.21 and Fig. 5.30). This occurs despite the initial Beltramian asymmetric axial wave amplitudes exceeding those of the complex-lamellar. The growth rate of the first unstable eigenvalue is notably larger for the complex-lamellar than the Beltramian models. This, along with the higher velocity mean flow of the Beltramian solutions, accounts for the accelerated degradation of the complex-lamellar flow.

Table 5.4: Backsubstitution and boundary condition error values for all plotted eigenvector examples.

Example	Equation				Boundary Conditions	
	Cont.	r-mom.	θ-mom.	z-mom.	$m(0)$	$m(1)$
Axisymmetric:						
Complex-Lamellar	8.33E-11	1.63E-10	2.38E-10	7.33E-10		
$u_r(r)$					0.00E+00	0.00E+00
$u_\theta(r)$					0.00E+00	0.00E+00
$u_z(r)$					3.24E-10	0.00E+00
$p(r)$					1.18E-10	5.89E-11
Linear Beltramian	1.48E-11	9.72E-11	1.14E-10	1.01E-10		
$u_r(r)$					0.00E+00	0.00E+00
$u_\theta(r)$					0.00E+00	0.00E+00
$u_z(r)$					8.90E-12	0.00E+00
$p(r)$					1.46E-10	2.47E-12
Harmonic Beltramian	1.36E-11	4.89E-11	1.69E-10	1.19E-10		
$u_r(r)$					0.00E+00	0.00E+00
$u_\theta(r)$					0.00E+00	0.00E+00
$u_z(r)$					7.56E-11	0.00E+00
$p(r)$					1.06E-10	1.16E-11
Asymmetric:						
Complex-Lamellar	2.03E-10	5.32E-10	7.08E-10	4.87E-10		
$u_r(r)$					0.00E+00	0.00E+00
$u_\theta(r)$					0.00E+00	0.00E+00
$u_z(r)$					0.00E+00	0.00E+00
$p(r)$					0.00E+00	7.60E-11
Linear Beltramian	9.85E-11	4.10E-10	4.63E-09	9.99E-10		
$u_r(r)$					0.00E+00	0.00E+00
$u_\theta(r)$					0.00E+00	0.00E+00
$u_z(r)$					0.00E+00	0.00E+00
$p(r)$					0.00E+00	3.32E-11
Harmonic Beltramian	1.73E-10	5.74E-10	3.77E-09	7.38E-10		
$u_r(r)$					0.00E+00	0.00E+00
$u_\theta(r)$					0.00E+00	0.00E+00
$u_z(r)$					0.00E+00	0.00E+00
$p(r)$					0.00E+00	2.85E-11

Chapter 6

Biglobal Stability Analysis of the Bidirectional Vortex

The previous chapter discussed the applicability of a one-dimensional stability analysis to the bidirectional vortex. This chapter is dedicated to the development of a general biglobal stability code that is suitable for the treatment of bidirectional vortex motion. At the outset, the deficiencies of the one-dimensional approach will be alleviated and a more precise understanding of bidirectional vortex turbulent breakdown will be achieved. The background work has been covered in Ch. 3, specifically Secs. 3.6.1–3.7. In these sections we laid the groundwork for the discretization and solution of PDEs via Chebyshev spectral collocation methods. In what follows, we will simply adapt our existing codes to solve the spectral form of Eqs. (A.9a–A.9d). The culmination of this chapter will consist of a complete two-dimensional, hydrodynamic stability code that is sufficiently generalized to handle any three-dimensional, axisymmetric, cylindrical base flow.

6.1 Deriving the Spectral Biglobal Equations

We consider the two-dimensional LNS equations as derived in App. A.2. These are repeated here for convenience:

Continuity:

$$\frac{\partial u_r}{\partial r} + \frac{u_r}{r} + iq\frac{u_\theta}{r} + \frac{\partial u_z}{\partial z} = 0 \tag{6.1a}$$

Radial momentum:

$$\begin{aligned} -i\omega u_r + U_r\frac{\partial u_r}{\partial r} + u_r\frac{\partial U_r}{\partial r} + iq\frac{U_\theta u_r}{r} + \frac{u_\theta}{r}\frac{\partial U_r}{\partial \theta} - \frac{2U_\theta u_\theta}{r} + U_z\frac{\partial u_r}{\partial z} + u_z\frac{\partial U_r}{\partial z} + \frac{\partial p}{\partial r} \\ = \varepsilon\left(\frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{u_r}{r^2} - \frac{q^2}{r^2}u_r - \frac{2iq}{r^2}u_\theta + \frac{\partial^2 u_r}{\partial z^2}\right) \end{aligned} \tag{6.1b}$$

Tangential momentum:

$$\begin{aligned} -i\omega u_\theta + U_r\frac{\partial u_\theta}{\partial r} + u_r\frac{\partial U_\theta}{\partial r} + iq\frac{U_\theta u_\theta}{r} + \frac{u_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r u_\theta}{r} + \frac{u_r U_\theta}{r} + U_z\frac{\partial u_\theta}{\partial z} + u_z\frac{\partial U_\theta}{\partial z} + iq\frac{p}{r} \\ = \varepsilon\left(\frac{\partial^2 u_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial u_\theta}{\partial r} - \frac{u_\theta}{r^2} - \frac{q^2}{r^2}u_\theta + \frac{2iq}{r^2}u_r + \frac{\partial^2 u_\theta}{\partial z^2}\right) \end{aligned} \tag{6.1c}$$

Axial momentum:

$$\begin{aligned} -i\omega u_z + U_r\frac{\partial u_z}{\partial r} + u_r\frac{\partial U_z}{\partial r} + iq\frac{U_\theta u_z}{r} + \frac{u_\theta}{r}\frac{\partial U_z}{\partial \theta} + U_z\frac{\partial u_z}{\partial z} + u_z\frac{\partial U_z}{\partial z} + \frac{\partial p}{\partial z} \\ = \varepsilon\left(\frac{\partial^2 u_z}{\partial r^2} + \frac{1}{r}\frac{\partial u_z}{\partial r} - \frac{q^2}{r^2}u_z + \frac{\partial^2 u_z}{\partial z^2}\right) \end{aligned} \tag{6.1d}$$

Our interest lies in the spectral decomposition and solution to this system given that ω is the circular frequency and an eigenvalue of the system. To discretize this system we follow the steps outlined in Sec. 3.6.3 and Sec. 5. The first builds a general outline that may be used, and the second extends the problem to a larger system. As usual we seek to formulate

the system in terms of the generalized eigenvalue problem:

$$A_{ij} f_i = \lambda B_{ij} f_i \tag{6.2}$$

To do so, we build the operator matrices, A_{ij} and B_{ij}, from smaller $N^2 \times N^2$ block matrices. Each block matrix refers to the operator of a specific dependent variable in one of the four governing equations. A general block decomposition is shown in Ch. 5 and given here as:

$$A_{ij} = \begin{array}{r} \\ \text{Cont.} \rightarrow \\ r-\text{mom.} \rightarrow \\ \theta-\text{mom.} \rightarrow \\ z-\text{mom.} \rightarrow \end{array} \begin{array}{c} \begin{array}{cccc} u_r(r) & u_\theta(r) & u_z(r) & p(r) \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \left[\begin{array}{c|c|c|c} A_{c,u_r} & A_{c,u_\theta} & A_{c,u_z} & A_{c,p} \\ \hline A_{r,u_r} & A_{r,u_\theta} & A_{r,u_z} & A_{r,p} \\ \hline A_{\theta,u_r} & A_{\theta,u_\theta} & A_{\theta,u_z} & A_{\theta,p} \\ \hline A_{z,u_r} & A_{z,u_\theta} & A_{z,u_z} & A_{z,p} \end{array}\right] \end{array}$$

$$B_{ij} = \begin{array}{r} \\ \text{Cont.} \rightarrow \\ r-\text{mom.} \rightarrow \\ \theta-\text{mom.} \rightarrow \\ z-\text{mom.} \rightarrow \end{array} \left[\begin{array}{c|c|c|c} B_{c,u_r} & B_{c,u_\theta} & B_{c,u_z} & B_{c,p} \\ \hline B_{r,u_r} & B_{r,u_\theta} & B_{r,u_z} & B_{r,p} \\ \hline B_{\theta,u_r} & B_{\theta,u_\theta} & B_{\theta,u_z} & B_{\theta,p} \\ \hline B_{z,u_r} & B_{z,u_\theta} & B_{z,u_z} & B_{z,p} \end{array}\right]$$

From this diagram we see that the final matrices are $4N^2 \times 4N^2$. This implies a significant increase in computing power to render solutions at the same resolution as the LNP approach where the matrices are only $4N \times 4N$. The factor of N in the biglobal matrices comes from the use of the Kronecker product to map spatial derivatives onto a product tensor grid. Fortuitously, the work by Chedevergne suggests that as much as an order of magnitude fewer collocation points can be used for each independent variable to converge to a viable solution [123]. This observation has yet to be verified.

At this point it is helpful to rewrite the governing equations in operator form. They can be expressed as

Continuity:

$$\left(\frac{\partial}{\partial r}+r^{-1}\right)u_r+\left(iqr^{-1}\right)u_\theta+\left(\boldsymbol{\frac{\partial}{\partial z}}\right)u_z=0 \tag{6.3a}$$

Radial momentum:

$$\begin{aligned}\left\{U_r\frac{\partial}{\partial r}+\frac{\partial U_r}{\partial r}+iqU_\theta r^{-1}+\boldsymbol{U_z\frac{\partial}{\partial z}}-\varepsilon\left[\frac{\partial^2}{\partial r^2}+r^{-1}\frac{\partial}{\partial r}-\left(1+q^2\right)r^{-2}+\boldsymbol{\frac{\partial^2}{\partial z^2}}\right]\right\}u_r\\ +\left(2iq\varepsilon r^{-2}-2U_\theta r^{-1}+r^{-1}\frac{\partial U_r}{\partial\theta}\right)u_\theta+\left(\frac{\partial U_r}{\partial z}\right)u_z+\left(\frac{\partial}{\partial r}\right)p=(i\omega)\,u_r\end{aligned} \tag{6.3b}$$

Tangential momentum:

$$\begin{aligned}\left(\frac{\partial U_\theta}{\partial r}+U_\theta r^{-1}-2iq\varepsilon r^{-2}\right)u_r+\left\{U_r\frac{\partial}{\partial r}+U_r r^{-1}+iqU_\theta r^{-1}+r^{-1}\frac{\partial U_\theta}{\partial\theta}+\boldsymbol{U_z\frac{\partial}{\partial z}}\right.\\ \left.-\varepsilon\left[\frac{\partial^2}{\partial r^2}+r^{-1}\frac{\partial}{\partial r}-\left(1+q^2\right)r^{-2}+\boldsymbol{\frac{\partial^2}{\partial z^2}}\right]\right\}u_\theta+\left(\frac{\partial U_\theta}{\partial z}\right)u_z+\left(iqr^{-1}\right)p=(i\omega)\,u_\theta\end{aligned} \tag{6.3c}$$

Axial momentum:

$$\begin{aligned}\left(\frac{\partial U_z}{\partial r}\right)u_r+\left(r^{-1}\frac{\partial U_z}{\partial\theta}\right)u_\theta+\left[U_r\frac{\partial}{\partial r}+iqU_\theta r^{-1}+\frac{\partial U_z}{\partial z}+\boldsymbol{U_z\frac{\partial}{\partial z}}\right.\\ \left.-\varepsilon\left(\frac{\partial^2}{\partial r^2}+r^{-1}\frac{\partial}{\partial r}-q^2r^{-2}+\boldsymbol{\frac{\partial^2}{\partial z^2}}\right)\right]u_z+\left(\boldsymbol{\frac{\partial}{\partial z}}\right)p=(i\omega)\,u_z\end{aligned} \tag{6.3d}$$

Clearly, these are very similar to the LNP equations. Here the axial dependence on the fluctuation is included by retaining derivatives of the fluctuations with respect to z. Their terms are bolded in the preceding equations. By appropriately applying the normal mode to the longitudinal derivatives, we see that the LNP system is actually a subset of these. This detail gives us confidence in our formulation thus far.

The domain must be transformed for both the r and z independent variables. For arbitrary chamber length, we consider the domain to be $0 \leq r \leq 1$ and $0 \leq z \leq Z_N$. The arbitrary specification of the location of the exit plane is necessary to properly transform the domain as well as generalize the formulation. We refer to Eq. (3.60) for a two-dimensional mapping formula. It gives

$$\begin{cases} r = \dfrac{1}{2}(\xi + 1) & \longleftrightarrow \quad \xi = 2r - 1 \\ z = \dfrac{Z_N}{2}(\eta + 1) & \longleftrightarrow \quad \eta = \dfrac{2z - Z_N}{Z_N} \end{cases} \quad \text{and,} \quad \begin{cases} \dfrac{\partial}{\partial r} = 2\dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial z} = \dfrac{2}{Z_N}\dfrac{\partial}{\partial \eta} \end{cases} \tag{6.4}$$

where $-1 \leq \xi \leq 1$ and $-1 \leq \eta \leq 1$ are required to use Chebyshev polynomial collocation.

At this point we are able to define the block matrices. We find

Continuity:

$$\begin{cases} A_{c,u_r} &= \bar{D}_N^r + r_{ii}^{-1} \\ A_{c,u_\theta} &= iqr_{ii}^{-1} \\ A_{c,u_z} &= \bar{D}_N^z \\ A_{c,p} &= 0 \end{cases} \quad \text{and} \quad \begin{cases} B_{c,u_r} &= 0 \\ B_{c,u_\theta} &= 0 \\ B_{c,u_z} &= 0 \\ B_{c,p} &= 0 \end{cases} \tag{6.5}$$

Radial momentum:

$$\left\{\begin{array}{lcl} A_{r,u_r} & = & U_{rii}\bar{D}^r_N + \left(\dfrac{\partial U_r}{\partial r}\right)_{ii} + iqU_{\theta ii}r_{ii}^{-1} + U_{zii}\bar{D}^z_N \\ & & -\varepsilon\left[(\bar{D}^r_N)^2 + r_{ii}^{-1}\bar{D}^r_N - (1+q^2)\,r_{ii}^{-2} + (\bar{D}^z_N)^2\right] \\ A_{r,u_\theta} & = & 2iq\varepsilon r_{ii}^{-2} - 2U_{\theta ii}r_{ii}^{-1} + r_{ii}^{-1}\left(\dfrac{\partial U_r}{\partial \theta}\right)_{ii} \\ A_{r,u_z} & = & \left(\dfrac{\partial U_r}{\partial z}\right)_{ii} \\ A_{r,p} & = & \bar{D}^r_N \end{array}\right. \quad \text{and} \quad \left\{\begin{array}{lcl} B_{r,u_r} & = & iI_N \\ B_{r,u_\theta} & = & 0 \\ B_{r,u_z} & = & 0 \\ B_{r,p} & = & 0 \end{array}\right. \tag{6.6}$$

Tangential momentum:

$$\left\{\begin{array}{lcl} A_{\theta,u_r} & = & \left(\dfrac{\partial U_\theta}{\partial r}\right)_{ii} + U_{\theta ii}r_{ii}^{-1} - 2iq\varepsilon r_{ii}^{-2} \\ A_{\theta,u_\theta} & = & U_{rii}\bar{D}^r_N + U_{rii}r_{ii}^{-1} + iqU_{\theta ii}r_{ii}^{-1} \\ & & +r_{ii}^{-1}\left(\dfrac{\partial U_\theta}{\partial \theta}\right)_{ii} + U_{zii}\bar{D}^z_N \\ & & -\varepsilon\left[(\bar{D}^r_N)^2 + r_{ii}^{-1}\bar{D}^r_N - (1+q^2)\,r_{ii}^{-2} + (\bar{D}^z_N)^2\right] \\ A_{\theta,u_z} & = & \left(\dfrac{\partial U_\theta}{\partial z}\right)_{ii} \\ A_{\theta,p} & = & iqr_{ii}^{-1} \end{array}\right. \quad \text{and} \quad \left\{\begin{array}{lcl} B_{\theta,u_r} & = & 0 \\ B_{\theta,u_\theta} & = & iI_N \\ B_{\theta,u_z} & = & 0 \\ B_{\theta,p} & = & 0 \end{array}\right. \tag{6.7}$$

Axial momentum:

$$\begin{cases} A_{z,u_r} &= \left(\dfrac{\partial U_z}{\partial r}\right)_{ii} \\ A_{z,u_\theta} &= r_{ii}^{-1}\left(\dfrac{\partial U_z}{\partial \theta}\right)_{ii} \\ A_{z,u_z} &= U_{rii}\bar{D}_N^r + iqU_{\theta ii}r_{ii}^{-1} + \left(\dfrac{\partial U_z}{\partial z}\right)_{ii} + U_{zii}\bar{D}_N^z \\ & \quad -\varepsilon\left[(\bar{D}_N^r)^2 + r_{ii}^{-1}\bar{D}_N^r - q^2 r_{ii}^{-2} + \bar{D}_N^z\right] \\ A_{z,p} &= \bar{D}_N^z \end{cases} \quad \text{and} \quad \begin{cases} B_{z,u_r} &= 0 \\ B_{z,u_\theta} &= 0 \\ B_{z,u_z} &= iI_N \\ B_{z,p} &= 0 \end{cases} \tag{6.8}$$

We remain consistent with LNP boundary conditions and those dictating an acoustically closed chamber. The radial conditions are identical to those in the LNP solution while the new streamwise boundary conditions can be drawn from the general acoustic boundary conditions, $\boldsymbol{n}\cdot\boldsymbol{u} = 0$ and $\boldsymbol{n}\cdot\nabla p = 0$, in the lateral direction. The work by Chedevergne [123] uses a two-dimensional streamfunction formulation with comparable, albeit slightly different, set of boundary conditions. Robitaillié-Montané and Casalis [124] develop a velocity-based formulation with a two-dimensional base flow in Cartesian coordinates. Their discussion is helpful, although it does not directly consider a three-dimensional mean flow. Finally, we determine that, for an eigenvalue problem in which the boundary conditions must be zero, the axisymmetric system must be closed by the boundary conditions and implemented in the respective code:

$$\begin{cases} u_r(0,z) &= 0 \quad \text{axisymmetry across the centerline} \\ u_r(1,z) &= 0 \quad \text{acoustically closed condition at the sidewall} \\ u_r(r,0) &= 0 \quad \text{no-slip at the headwall} \\ u_r(r,Z_f) &= 0 \quad \text{no-slip across the acoustic boundary} \end{cases} \tag{6.9}$$

```
% ur(0,z) and ur(1,z)
Amat(Arbc,j) = II(Arbc,:);              % The IV operator on ur from BC on ur
Bmat(Arbc,j) = 0*II(Arbc,:);            % The IV operator on ur from BC on ur
Amat(Brbc,j) = II(Brbc,:);              % The BC operator on ur from BC on ur
Bmat(Brbc,j) = 0*II(Brbc,:);            % The BC operator on ur from BC on ur

% ur(r,0) and ur(r,Zf)
Amat(Azbc,j) = II(Azbc,:);              % The IV operator on ur from BC on ur
Bmat(Azbc,j) = 0*II(Azbc,:);            % The IV operator on ur from BC on ur
Amat(Bzbc,j) = II(Bzbc,:);             % The BC operator on ur from BC on ur
Bmat(Bzbc,j) = 0*II(Bzbc,:);            % The BC operator on ur from BC on ur
```

$$\begin{cases} u_\theta(0,z) & = & 0 & \text{axisymmetry across the centerline} \\ u_\theta(1,z) & = & 0 & \text{no-slip at the sidewall} \\ u_\theta(r,0) & = & 0 & \text{no-slip at the headwall} \\ u_\theta(r,Z_f) & = & 0 & \text{no-slip across the acoustic boundary} \end{cases} \tag{6.10}$$

```
% uq(0,z) and uq(1,z)
Amat(N^2+Arbc,N^2+j)=II(Arbc,:);        % The IV operator on uq from BC on uq
Bmat(N^2+Arbc,N^2+j)=0*II(Arbc,:);      % The IV operator on uq from BC on uq
Amat(N^2+Brbc,N^2+j)=II(Brbc,:);        % The BC operator on uq from BC on uq
Bmat(N^2+Brbc,N^2+j)=0*II(Brbc,:);      % The BC operator on uq from BC on uq

% uq(r,0) and uq(r,1)
Amat(N^2+Azbc,N^2+j)=II(Azbc,:);        % The IV operator on uq from BC on uq
Bmat(N^2+Azbc,N^2+j)=0*II(Azbc,:);      % The IV operator on uq from BC on uq
Amat(N^2+Bzbc,N^2+j)=II(Bzbc,:);       % The BC operator on uq from BC on uq
Bmat(N^2+Bzbc,N^2+j)=0*II(Bzbc,:);      % The BC operator on uq from BC on uq
```

$$\begin{cases} \partial_r u_z(0,z) & = & 0 & \text{axisymmetry across the centerline} \\ u_z(1,z) & = & 0 & \text{no-slip at the sidewall} \\ u_z(r,0) & = & 0 & \text{acoustically closed condition at the headwall} \\ u_z(r,Z_f) & = & 0 & \text{acoustically closed condition at the endwall} \end{cases} \tag{6.11}$$

```
% uz(0,z) and uz(1,z)
Amat(2*N^2+Arbc,N^2+j)=0*DDr(Arbc,:);  % The IV operator on uq from BC on uz
Bmat(2*N^2+Arbc,2*N^2+j)=0*II(Arbc,:); % The IV operator on uz from BC on uz
Amat(2*N^2+Brbc,2*N^2+j)=II(Brbc,:);   % The BC operator on uz from BC on uz
Bmat(2*N^2+Brbc,2*N^2+j)=0*II(Brbc,:); % The BC operator on uz from BC on uz

% uz(r,0) and uz(r,1)
Amat(2*N^2+Azbc,2*N^2+j)=II(Azbc,:);   % The IV operator on uz from BC on uz
Bmat(2*N^2+Azbc,2*N^2+j)=0*II(Azbc,:); % The IV operator on uz from BC on uz
Amat(2*N^2+Bzbc,2*N^2+j)=II(Bzbc,:);  % The BC operator on uz from BC on uz
Bmat(2*N^2+Bzbc,2*N^2+j)=0*II(Bzbc,:); % The BC operator on uz from BC on uz
```

$$\begin{cases} \partial_r p(0,z) & = & 0 & \text{axisymmetry across the centerline} \\ \partial_r p(1,z) & = & 0 & \text{acoustically closed condition at the sidewall} \\ \partial_z p(r,0) & = & 0 & \text{acoustically closed condition at the headwall} \\ \partial_z p(r,Z_f) & = & 0 & \text{acoustically closed condition at the endwall} \end{cases} \tag{6.12}$$

```
% p(0,z) and p(1,z)
Amat(3*N^2+Arbc,3*N^2+j)=DDr(Arbc,:);  % The IV operator on p from BC on p
Bmat(3*N^2+Arbc,3*N^2+j)=0*II(Arbc,:); % The IV operator on p from BC on p
```

```
Amat(3*N^2+Brbc,3*N^2+j)=DDr(Brbc,:);   % The BC operator on p from BC on p
Bmat(3*N^2+Brbc,3*N^2+j)=0*II(Brbc,:); % The BC operator on p from BC on p

% p(r,0) and p(r,1)
Amat(3*N^2+Azbc,3*N^2+j)=DDz(Azbc,:);   % The IV operator on p from BC on p
Bmat(3*N^2+Azbc,3*N^2+j)=0*II(Azbc,:); % The IV operator on p from BC on p
Amat(3*N^2+Bzbc,3*N^2+j)=DDz(Bzbc,:);   % The BC operator on p from BC on p
Bmat(3*N^2+Bzbc,3*N^2+j)=0*II(Bzbc,:); % The BC operator on p from BC on p
```

Recall from Ch. 3 that `Arbc, Brbc, Azbc, Bzbc` are vectors designed to track the matrix elements at the boundaries after the product tensor grid is generated. They are necessary since the boundary elements do not appear on the outermost rows and columns as in the LNP formulation. The streamwise conditions on all velocities at the headwall are zero as a means of satisfying no slip. At the endwall, the boundary conditions on velocity and pressure are selected to represent an acoustically closed chamber. This is a difference between the streamfunction and velocity-based formulations. An acoustically closed boundary condition appears to be physically consistent with a choked rocket chamber. The reader is cautioned that the code given is not the complete implementation of the boundary conditions and is included as an example only. In actuality, whenever a specific condition is imposed on one variable, the operators acting on the other dependent variables must be set to zero.

The asymmetric analog is given as:

$$\begin{cases} u_r(0,z) &= 0 \\ u_r(1,z) &= 0 \\ u_r(r,0) &= 0 \\ u_r(r,Z_f) &= 0 \end{cases} \qquad \begin{cases} u_\theta(0,z) &= 0 \\ u_\theta(1,z) &= 0 \\ u_\theta(r,0) &= 0 \\ u_\theta(r,Z_f) &= 0 \end{cases} \tag{6.13}$$

$$\begin{cases} \boldsymbol{u_z(0,z)} &= \mathbf{0} \\ u_z(1,z) &= 0 \\ u_z(r,0) &= 0 \\ u_z(r,Z_f) &= 0 \end{cases} \qquad \begin{cases} \boldsymbol{p(0,z)} &= \mathbf{0} \\ \partial_r p(1,z) &= 0 \\ \partial_z p(r,0) &= 0 \\ \partial_z p(r,Z_f) &= 0 \end{cases} \tag{6.14}$$

The asymmetric boundary conditions are the same as the axisymmetric except the for axial velocity and pressure fluctuations that must now exhibit a node point to allow for higher tangential modes to exist. We can implement this scenario by simply adding the conditional statement:

```
if q≠0
    Amat(2*N^2+Arbc,2*N^2+j)=II(Arbc,:); % The IV operator on uz from BC on uz
    Amat(3*N^2+Arbc,3*N^2+j)=II(Arbc,:); % The IV operator on p from BC on p
end
```

6.2 On the Hardware Requirements

The hardware requirements are significantly higher for the biglobal approach than the LNP. Unlike the LNP operator matrices, those for the BG approach require the conversion into product tensor form. Each equation has four operator submatrices: one for each dependent variable. Rather than a complete operator matrix of size $4N \times 4N$ for the LNP approach,

the biglobal operator matrices are $4N^2 \times 4N^2$ because the use of Kronecker products to map the spectral derivatives to a product tensor grid increases both rows and columns by a factor of N. According to Theofilis, we can expect at least 30 to 35 discretization points (N) for each fluctuating variable in the governing equation to adequately describe the spatial structure of the eigenfunction [41]. The degree of computational power required to handle approximately 40-50 points is at the edge of what current desktop computers are capable of supporting. Furthermore, Fig. 6.1 suggests that this level of refinement is still insufficient. Some computing load can be alleviated through the use of a streamfunction formulation for a two-dimensional solution. Unfortunately, the general formulation presented here cannot take advantage of a strictly two-dimensional streamfunction formulation while still accommodating a three-dimensional perturbation and mean flow. Furthermore, more primitive formulations such as the Rayleigh equation that ignores viscosity offer little relief when resolved using the biglobal approach. This is true since the total number of variables remain unchanged and it is this that controls the size of the matrices. The only gain associated with an inviscid formulation is a potential reduction in the number of collocation points required to resolve the domain in the absence of thin boundary layers. It can potentially reduce the overall required discretization points since it neglects boundary layers. This route may be ill-advised given the influence of shear layer vorticity generation and its effect on flow breakdown.

The bulk of the computational load is housed in the eigensolver. Traditional QZ and LZ algorithms compute the entire spectrum but are much more time consuming than Krylov subspace methods. In this vein, Arnoldi algorithms are typically substituted. Arnoldi methods compute a specified number of eigenvalues located around a guess value. A common way of determining guess values is to revert to low resolution QZ or LZ algorithms. This is a potentially pitfall. It is shown in Fig. 5.1 and Fig. 6.1 that grid refinement is necessary to determine the minimum number of collocation points for convergence. Below the minimum number of collocation points, the spectrum shows significant dependence N. Figure 6.1

Table 6.1: The minimum discretization number, N, to characterize the base flow boundary layer with at least three points and estimated runtime for the QZ eigensolver. All times estimated for an Intel Core2 CPU Quad 6600 @ 2.4GHz with 4 Gb RAM and Win 7x64.

	$Re =$	1000	5000	10000	50000	100000
	$V =$	628	3142	6283	31416	62832
Complex Lamellar						
	$\delta_w =$	0.0230	0.0046	0.0023	0.0005	0.0002
	$N \approx$	21	44	57	150	200
	estimated runtime (hrs)	0.04	20	-	-	-
Beltramian						
	$\delta_w =$	0.0094	0.0019	0.0009	0.0002	0.0001
	$N \approx$	40	80	100	250	350
	estimated runtime (hrs)	6	-	-	-	-

does not show the precise convergence achieved with the one-dimensional approach. Due to computational and run-time limitations, a complete parametric study with significantly more collocation points is unfeasible at the present. Nonetheless, even though the spectrum appears to lack convergence, the waveforms predicted by the eigenfunctions are convergent.

To further exacerbate the need for a fine spectral grid, we realize that high Reynolds numbers lead to an increasingly diminished boundary layer thickness. The collocation near the wall must be such that the correct boundary layer profile is realized. Poor spacing within the boundary layer equates to an inaccurate profile and, more significantly, incorrect characterization of the shear stresses generated by the base flow. Large Reynolds numbers can also cause unphysical oscillations in the interpolating polynomial due to the steepness of the boundary layer profile and inability to achieve proper collocation. Table 6.1 approximates the minimum collocation number to characterize the boundary layer with at least three points. Actual case studies require significantly more RAM to complete. Omitted cells in Table 6.1 indicate where a typical high performance desktop is no longer able to complete the study.

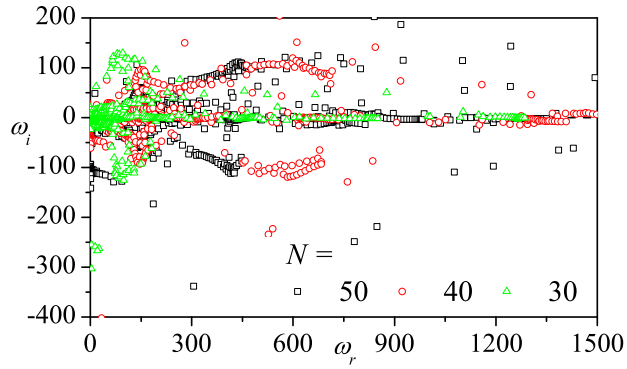
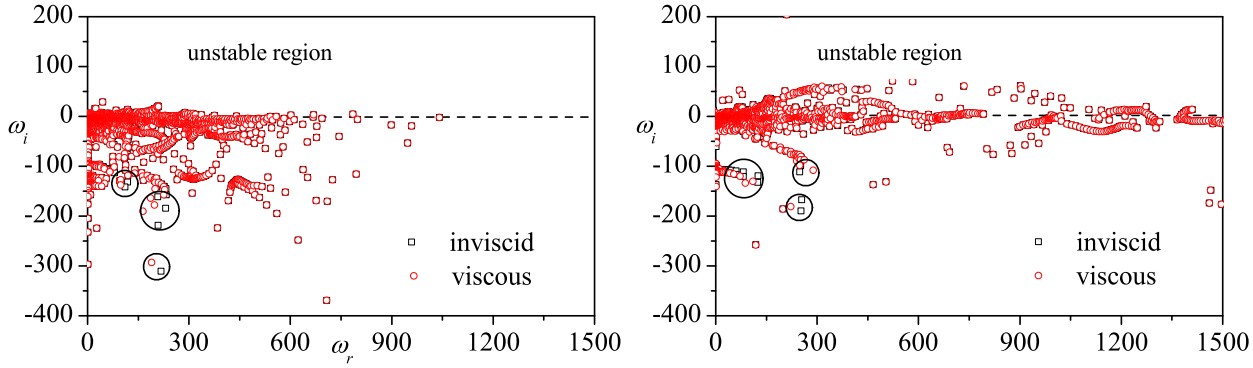


Figure 6.1: Qualitative grid refinement using the linear Beltramian model with $l = 2$, $Re = 10000$, $\kappa = 0.1$ and $q = 1$.



a) $Re = 2000$ **b)** $Re = 10000$

Figure 6.2: Comparison of the spectrum for the inviscid versus the viscous complex-lamellar solution with $q = 1$, $l = 2$, and $\kappa = 0.1$.

In order to assess the influence of viscosity on our base flow model, Fig. 6.2 is used to compare the spectra predicted by the biglobal instability framework using both inviscid and viscous mean flow representations at two values of the Reynolds number. Due to the overlap between inviscid and viscous eigenvalues in Fig. 6.2, it is clear that the inclusion of viscosity in the base flow has a negligible effect on the spectral results to the extent that the inviscid and viscous spectra are nearly indiscernible. Sporadic exceptions are identified by circles in Fig. 6.2. The two values of the Reynolds number are considered to illuminate the aforementioned problem of grid spacing for a diminishing boundary layer thickness (Table

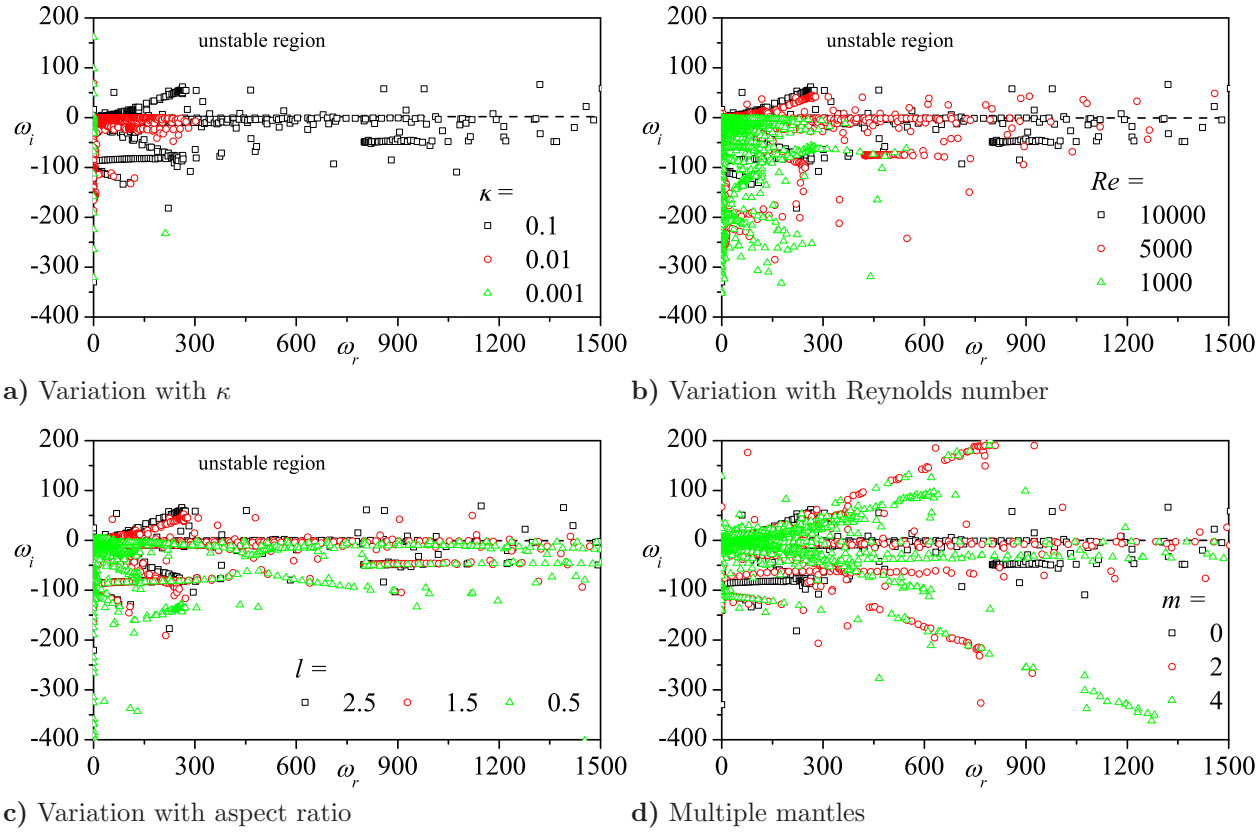
6.1). We see that even though Fig. 6.2b considers a higher Reynolds number that falls outside of the acceptable range, no appreciable difference may be identified. Also, the inverse is true for Fig. 6.2a. Similar findings were shown for the one-dimensional analysis in Ch. 5. These results suggest that the inclusion of a sidewall shear layer in the mean flow may not be necessary to characterize the hydrodynamic stability.

6.3 The Complex-Lamellar Bidirectional Vortex

The following results are for the viscous complex-lamellar bidirectional vortex. The trigonometric composition of the base flow separates itself from the two Beltramian solutions. Although similarities can and should be expected, there will likely be definite differences as well. For reference, the LNP results can be found in Ch. 5.

6.3.1 Axisymmetric Spectrum

Figure 6.3a shows very small variation with κ. It also illustrates very "straight-line" spectral structures, most of which occurring horizontally and very near the critical line of $\omega_i = 0$. A much greater disparity occurs when varying the Reynolds number seen in Fig. 6.3b. Specifically, we see a shift in the overall spectrum toward damped eigenvalues with a decrease in Re. This is to be expected given that increases in Re correlate to smaller diffusive viscosity and, more practically, higher velocity and more energetic injection. Continuing with Fig. 6.3, we see in Fig. 6.3c that only subtle differences exist between chambers of different aspect ratio. The figure suggests that shorter aspect ratio chambers entail smaller amplification of undamped eigenvalues at similar circular frequencies. Overall, differences between these spectral results are small for the majority of eigenvalues. One thing to notice, however, is that small aspect ratio chambers ($0 \leq l \leq 1.5$) share spectral structures that larger aspect ratios ($1.5 \leq l \leq 2.5$) do not exhibit and vice versa. Lastly, there are very apparent changes



a) Variation with κ

b) Variation with Reynolds number

c) Variation with aspect ratio

d) Multiple mantles

Figure 6.3: Axisymmetric parametric study for several input parameters. Here $q = 0$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

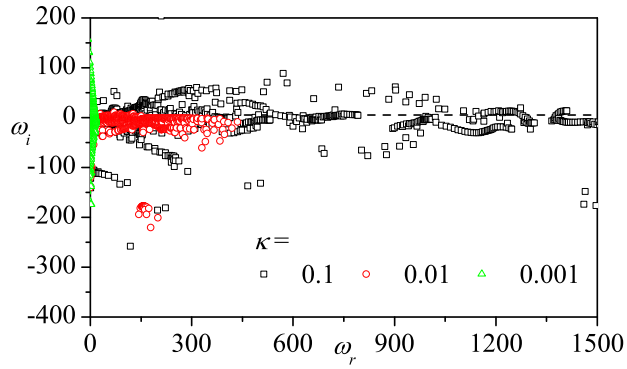
in the spectrum for higher mode, multidirectional flows seen in Fig. 6.3d. Though the straight line spectral structures are still apparent, increases in the number of flow reversals extend these structures to higher and higher circular frequencies. Many of these higher frequency eigenvalues are accompanied by larger amplification and damping as well.

Although each of the studies in Fig. 6.3 show amplified eigenvalues, their spectral results correspond to zero or very near-zero eigenvectors in all directions. This is a very interesting result for a variety of reasons. The first being that axisymmetric modes are seemingly unimportant for this particular flowfield. This is not the case for the LNP results shown in Fig. 5.8. Those results suggest that the amplified waveforms are actually large and three dimensional. Clearly we are seeing a significant difference between the

two results. Moreover, past research has concluded that the axisymmetric mode is most relevant. This is clearly not the case shown in these results. We cannot, however, conclude that the omission of asymmetric modes is incorrect given that the vast majority of work in one and two-dimensional hydrodynamic instability considers two-dimensional base flows and streamfunction representations of the stability equations. In those formulations the axisymmetric fluctuations may be sufficient; however, the zero amplitude eigenvectors computed here for the first tangential mode are not sufficient to characterize hydrodynamic instability for swirl driven base flows.

6.3.2 Asymmetric Spectrum

The distribution of eigenmodes for $0.001 \leq \kappa \leq 0.1$ is illustrated in Fig. 6.4 where small disparities may be noted at different values of κ. Both undamped and damped modes may be seen although the most undamped modes correspond to large values of κ. For all the cases considered, the eigenvalues circulate near and around the abscissa and are less sensitive to variations in κ than Re. On the other hand, Fig. 6.5 shows a larger difference in the spectrum with regards to the Reynolds number. The low Reynolds number case exhibits the highest undamped eigenvalues near the origin but does not extend into the higher frequency region as the larger Reynolds number cases do. In general, we see the larger the Reynolds number gets, the farther into the high frequency domain the spectrum persists. Arguably, $\kappa = 0.01$ may be a more physical choice. This parameter quantifies the relative size between the tangential and the axial and radial velocities by giving a measure of the swirl intensity. As κ decreases, the swirl intensity increases. This means the overall flowfield becomes more swirl dominated. One could expect greater swirl would stabilize the flow through increased centrifugal forces acting to inhibit vortex generation. For this reason, it is fruitful to consider our parametric study for two values of κ that encloses the set of physical parameters. Figure 6.5b reconsiders the effect of varying Re for $\kappa = 0.01$. The differences are immediately evident. First, we



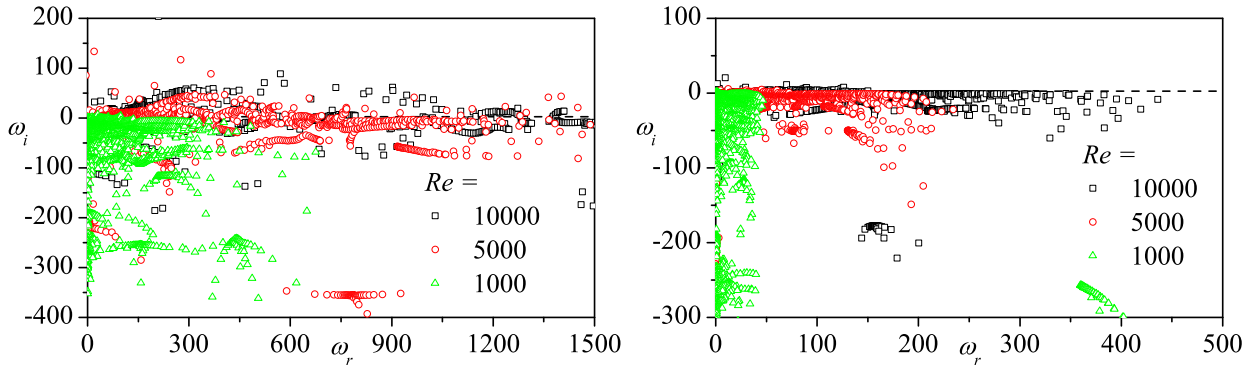
a) Variation with κ

Figure 6.4: Asymmetric parametric study for several values of κ. Here $q = 1$, $l = 2$, and $Re = 10,000$.

observe that the spectral results are confined to lower frequencies. Furthermore, nearly all predicted eigenvalues fall below the stability line. The growth of those few outliers appearing in the undamped region is coupled by many highly damped eigenvalues appearing at the same or similar circular frequencies. The overall effect is a stable prediction for all values of Re.

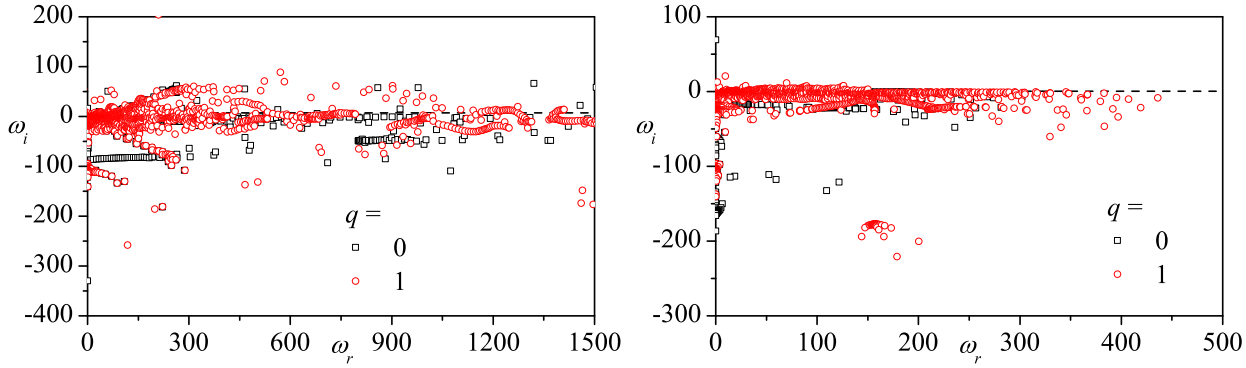
Axisymmetric modes have a slightly different spectral character than higher mode numbers. Figure 6.6a shows a general overlap of the spectrum in the low frequency eigenvalues for mode numbers greater than zero. This overlap is not so apparent at higher frequency. No definitive statement can be made about the difference in stability between mode numbers. Very little difference can be seen between the two tangential mode numbers when κ is decreased in Fig. 6.6b. Again, most eigenvalues appear below the critical line and those appearing above are damped by stable eigenvalues at similar circular frequencies.

Figure 6.7 depicts a typical waveform of each velocity component with $\kappa = 0.1$. Here, the graphs depict contour plots representing the magnitude of u_r, u_θ, u_z, and p for the first unstable eigenvalue. If we recall Fig. 2.3, we can make a key observation about the stability of the complex-lamellar vortex. The first should be that the unmodified amplitude of the hydrodynamic wave is of the same order as the base flow itself for the axial and radial components. In fact, the maximum value of the radial perturbation exceeds that of the base



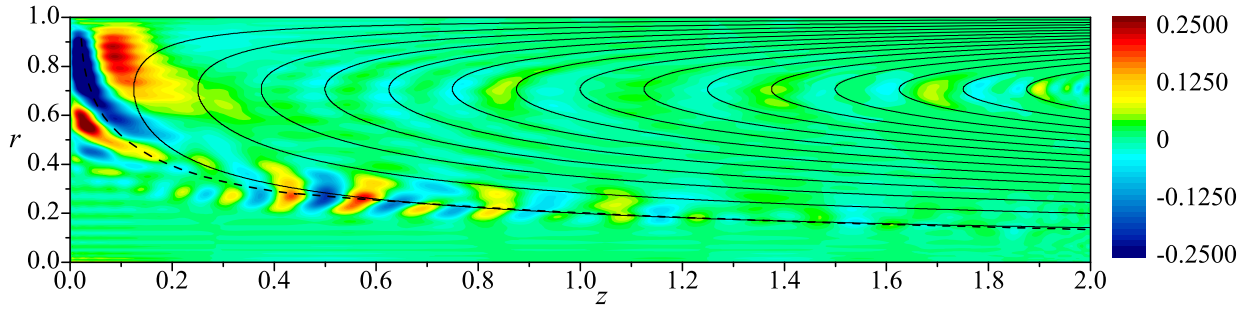
a) Variation with Reynolds number, $\kappa = 0.1$ **b)** Variation with Reynolds number, $\kappa = 0.01$

Figure 6.5: Asymmetric parametric study for several input parameters. Here $q = 1$, $l = 2$, and $Re = 10,000$.

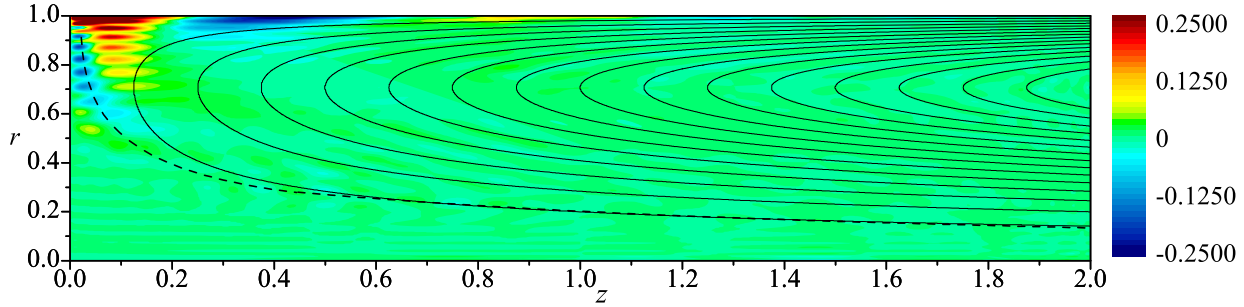


a) Variation with tangential mode number, $\kappa = 0.1$ **b)** Variation with tangential mode number, $\kappa = 0.01$

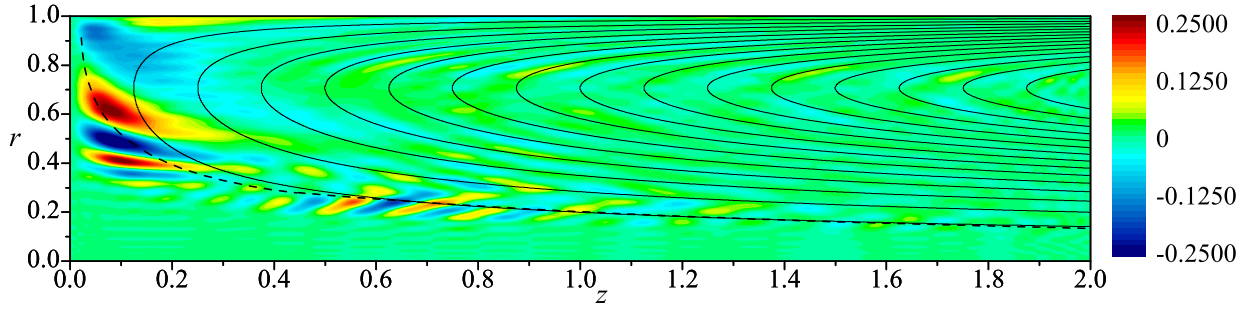
Figure 6.6: Spectral results for higher mode numbers. Here $l = 2$ and $Re = 10,000$.



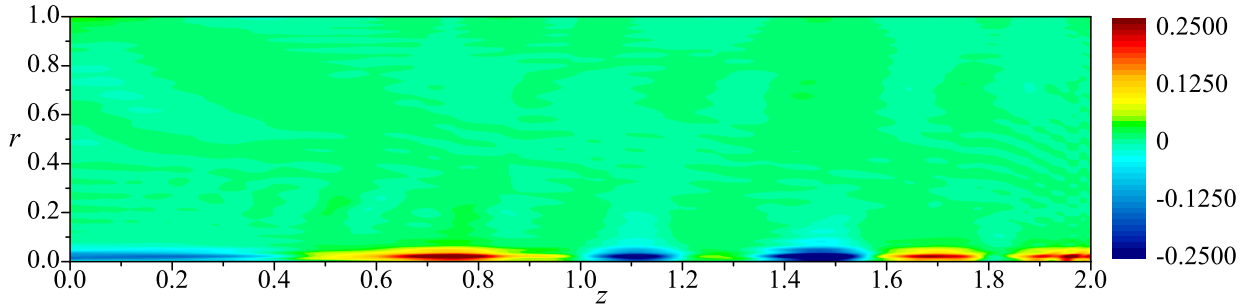
a) Biglobal radial velocity wave for the complex-lamellar model



b) Biglobal tangential velocity wave for the complex-lamellar model

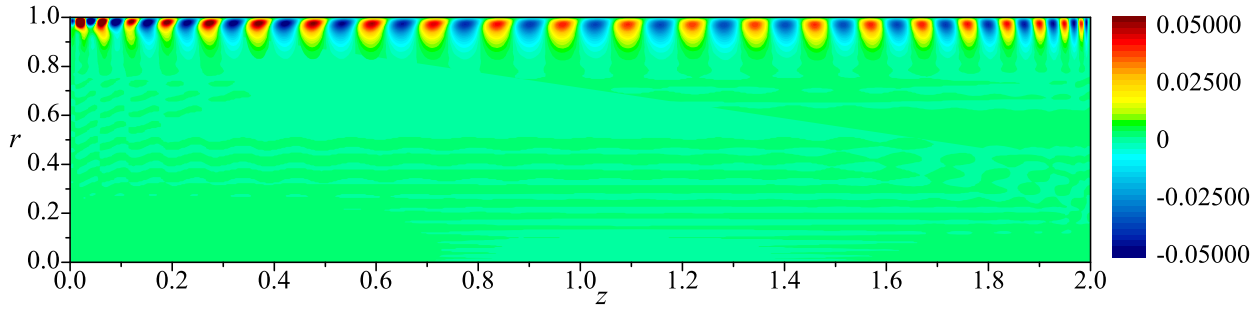


c) Biglobal axial velocity wave for the complex-lamellar model

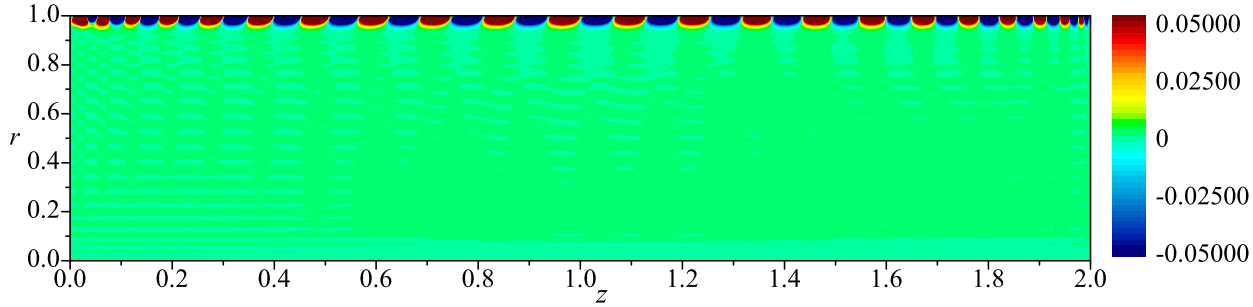


d) Biglobal pressure wave for the complex-lamellar model

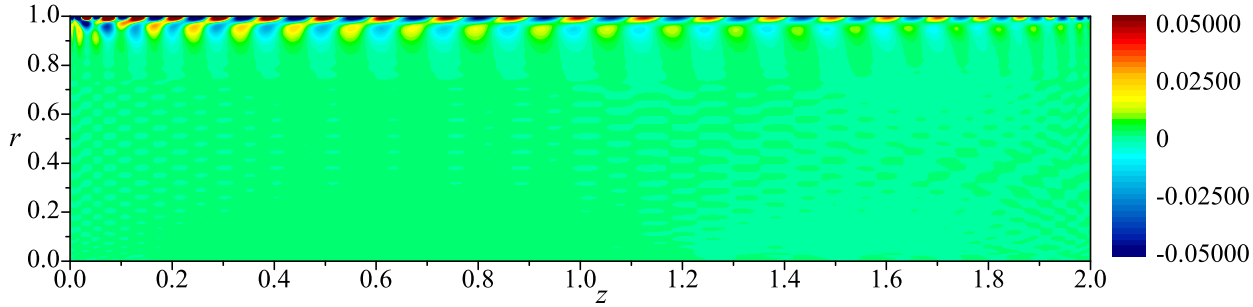
Figure 6.7: Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.2178 + 0.2940i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. See Table 6.2 for error values.



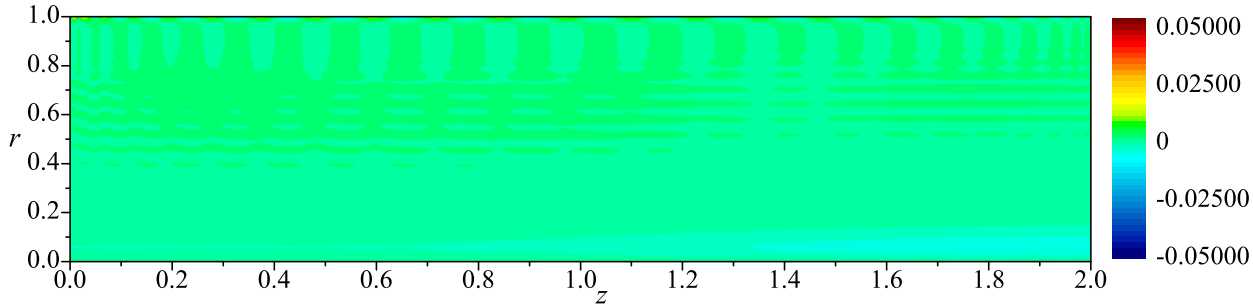
a) Biglobal radial velocity wave for the complex-lamellar model



b) Biglobal tangential velocity wave for the complex-lamellar model



c) Biglobal axial velocity wave for the complex-lamellar model



d) Biglobal pressure wave for the complex-lamellar model

Figure 6.8: Asymmetric eigensolutions for the unstable eigenvalue, $0.3443 + 0.4257i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.01$.

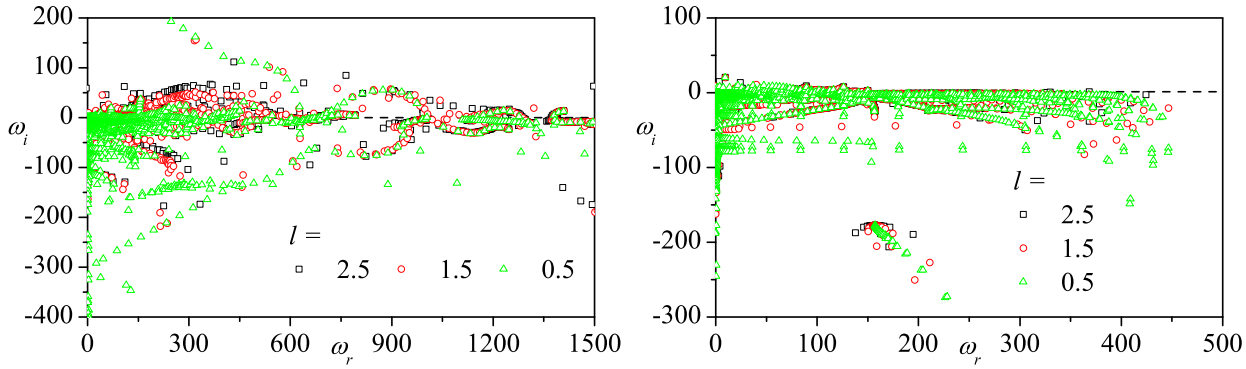
flow. This suggests that even without amplification, this flowfield will have a propensity toward turbulent breakdown with the possibility of destroying the coherent bidirectional motion completely.

When comparing each waveform against the others we witness consistent flow patterns for each perturbation. For instance largest amplitude waves occur near the headwall; however, the radial position of maximum oscillation varies in each direction. Perhaps the most interesting characteristic is the apparent oscillations around the streamlines of the base flow shown by the solid black lines. Recall that this analysis was not constructed from a streamfunction approach, yet the oscillation contours most strikingly follow the streamlines throughout the majority of the chamber length. This feature is seen in all three components, though not always present in the pressure profile. It does however, only show up for high Reynolds number studies. At low Re, the waveforms look much similar to those in the LNP approach.

In general, the pressure perturbation hovers very near zero for the majority of the chamber. From this, we can conclude that the instantaneous pressure is nearly unmodified with the exception of a thin region in which the pressure oscillates along the streamwise direction very near the centerline. This region corresponds to the largest gradient in the base flow pressure profile. Even so, the average over the entire domain is nearly zero.

Lastly, we note that a funnel shape is apparent near the centerline. The boundary of the funnel is identified in Fig. 6.7 by the dashed line. This indicates a somewhat stable region where the outer vortex spills inwardly to become the inner vortex. This is consistent with early speculation as to the shape of the headwall boundary layer.

When κ is reduced, the tangential velocity becomes the dominant contributor to the wave form. The contours in Fig. 6.8 no longer show oscillations about the streamlines as in the previous case. Rather, the highest amplitudes appear in the wall region and are attenuated. This indicates that the tangential velocity is hindering wave formation throughout the majority of the chamber and centrifugal forces are confining oscillations near



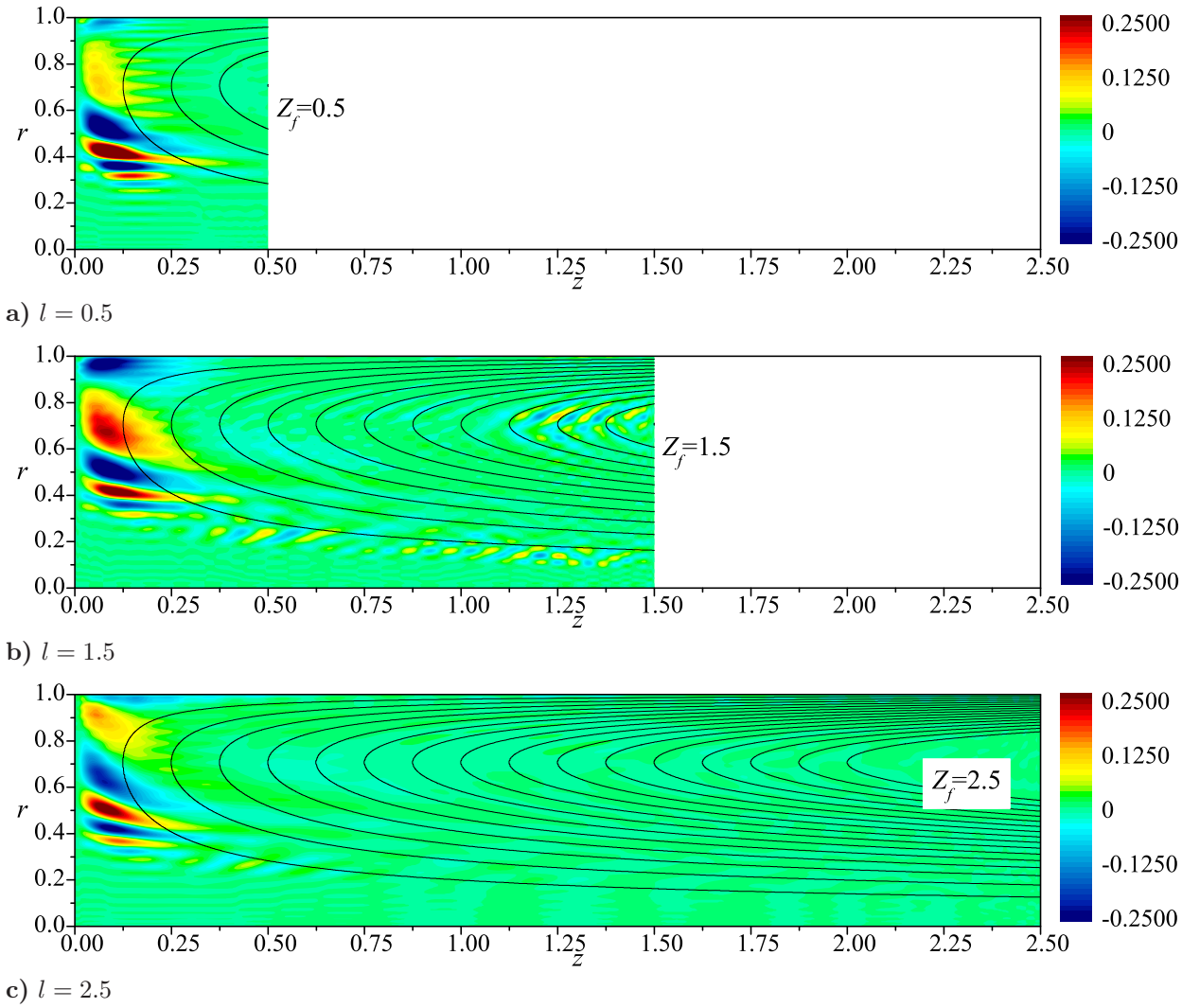
a) Variation with aspect ratio, $\kappa = 0.1$ **b)** Variation with aspect ratio, $\kappa = 0.01$

Figure 6.9: Asymmetric eigenvalues for several aspect ratios. Here $q = 1$ and $Re = 10,000$.

the sidewall. This is distinctly different from the previous contour plots where the maximum values of the axial and radial velocities were appreciable compared to the tangential velocity.

Changing the Aspect Ratio

Changing the aspect ratio is similar to considering different axial positions in the LNP analysis. Figure 6.9 illustrates the effect of variable l on the spectrum. Upon careful examination, we can suggest that the smallest aspect ratio considered, $l = 0.5$, shows a unique pattern when compared to the other test cases. This pattern is identified by the undamped pseudo-continuous spectral line ranging from about $\omega = 200 + 200i$ to $\omega = 750 + 0i$. This spectral pattern is reflected about the line $\omega_i = 0$. While eigenvalues resulting from other chamber lengths fall on or near this line, it is distinctly constructed from the spectrum for $l = 0.5$. The four other test cases show very small variations in their spectral results. Each case is characterized by similar spectral structures and pseudo-continuous spectral lines. This plot seems to suggest that the small aspect ratio chamber will be more susceptible to temporal instability than its longer counterparts. Given the unique spectral structures for $l = 0.5$, it could also be insinuated that the stability of chambers with aspect ratios less than one may behave differently than those that are more slender.



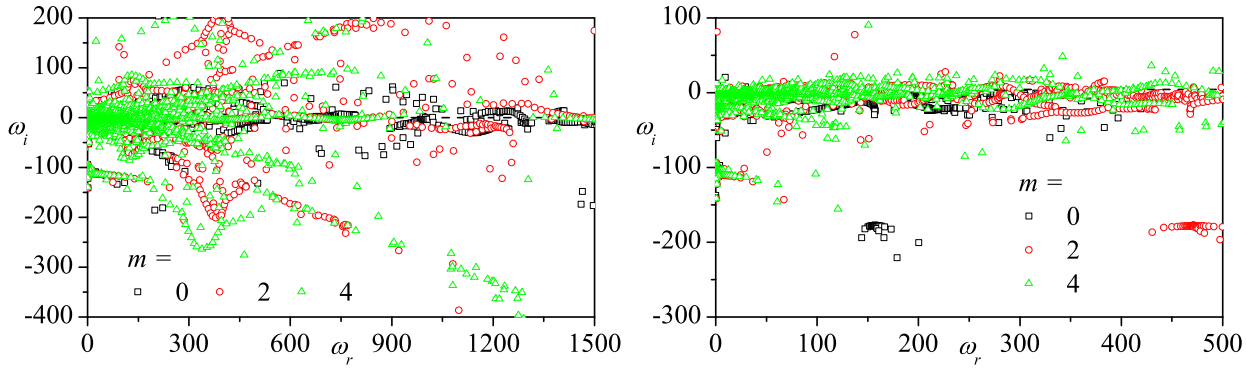
a) $l = 0.5$

b) $l = 1.5$

c) $l = 2.5$

Figure 6.10: Axial waveform of the first unstable eigenvalue of the complex-lamellar bidirectional vortex for different aspect ratios with $q = 1$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$.

Conversely, when $\kappa = 0.01$, very little difference between aspect ratios is seen in the spectral plots. The aspect ratio seems to be uninfluential when compared to other test parameters when κ is small. By considering the overall growth at discrete circular frequencies we find that when $\omega_r = 90$ the average growth rate returns a slightly undamped value of 0.0428 for $l = 2.5$, indicating slow linear growth about that frequency. This is the only parameter combination that predicts positive growth in this figure. For comparison, when



a) Variation with multiple mantles, $\kappa = 0.1$ **b)** Variation with multiple mantles, $\kappa = 0.01$

Figure 6.11: Asymmetric parametric study for multi-directional flow. Here $q = 1$, $l = 2$, and $Re = 10,000$.

$\kappa = 0.1$, many (100+) discrete frequencies can be expected to be unstable. Again, the stabilizing effect of an increased swirl number is evident.

The axial waveform sheds more light on the question of aspect ratio. Figure 6.10 shows contour plots at three increasing aspect ratios. All show the region of largest instability located near the headwall and significantly reduced wave amplitudes following the lateral direction. While the high amplitude region is a significant portion of the small aspect ratio chamber, this region appears to be drawn downstream as the chamber lengthens. Cross-sections near the headwall show a similar oscillatory behavior with a slight variation to the radial shape of the axial wave.

Multiple Mantles

As previously discussed, the existence of multidirectional flow is a definite possibility when considering only the base flow. From physical speculations, we could assume that increasing the number of flow reversals would increase the effect of hydrodynamic instability. This effect could be in the form of higher amplitudes, more destructive waveforms, or both.

With respect to the possibility of higher amplitudes, Fig. 6.11 shows that higher amplitude waves should be expected for successive increases in flow reversals. Similar spectral

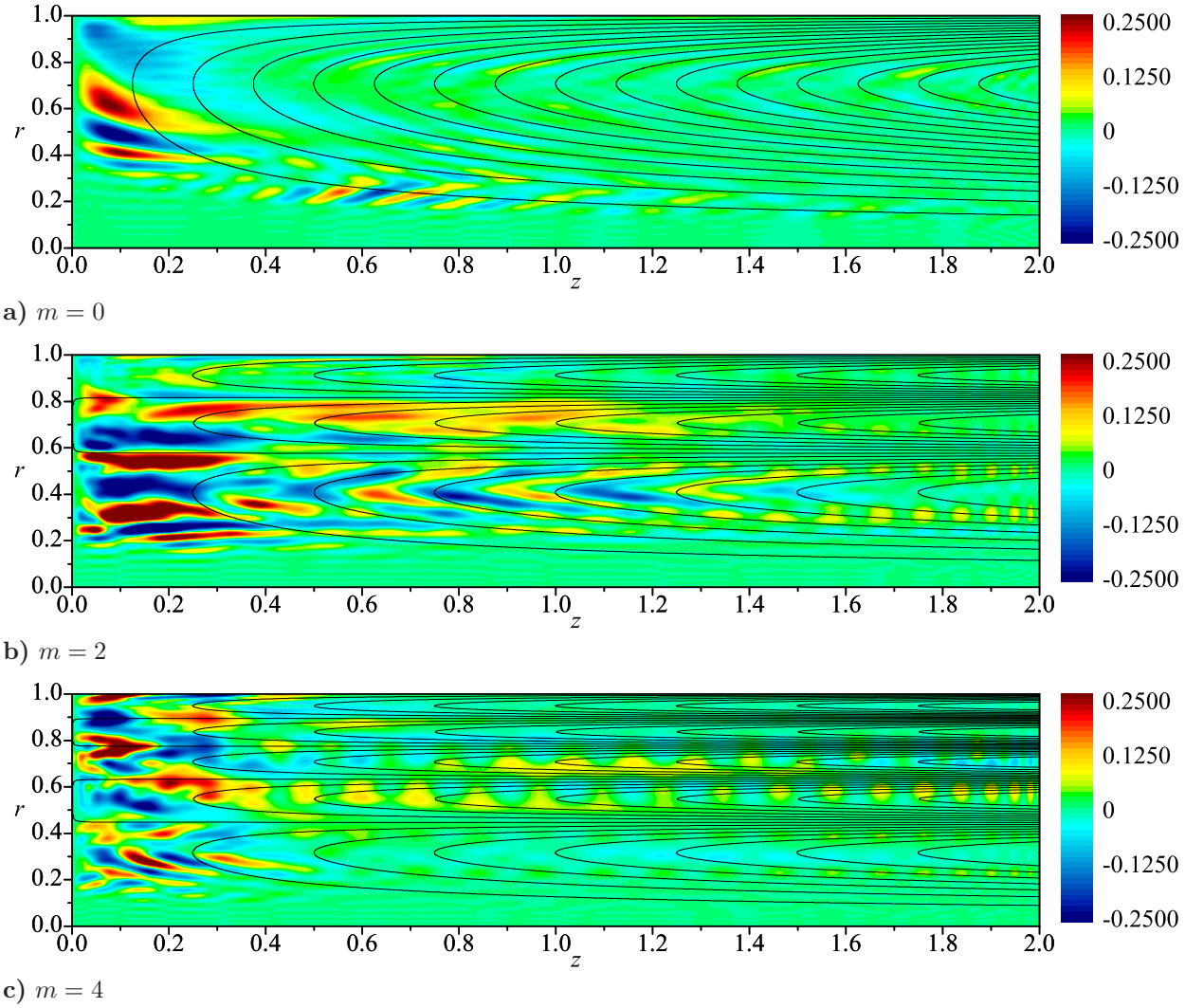


Figure 6.12: Axial waveform of the first unstable eigenvalue of the complex-lamellar bidirectional vortex for multidirectional flow with $q = 1$, $Re = 10,000$, and $\kappa = 0.1$.

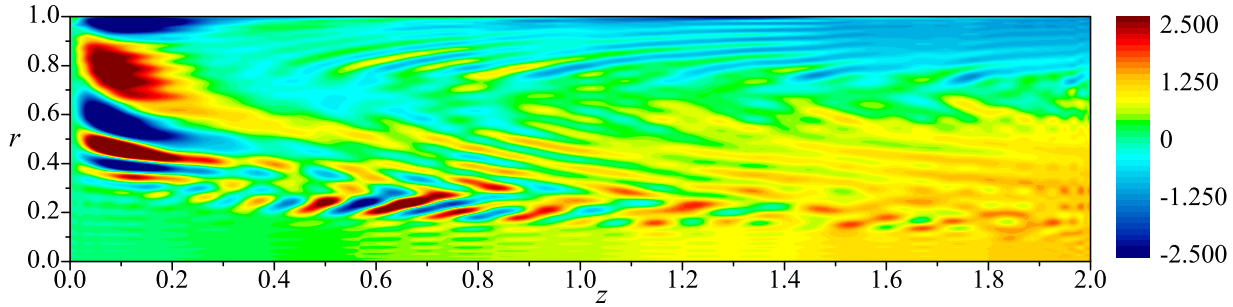
structures appear in both the $m = 2$ and $m = 4$ cases with the apparent shift toward both higher damped and undamped waves. To complement the increased amplitudes, the waveforms shown in Fig. 6.12 are significantly more disheveled than those for bidirectional flow. Oscillations continue to occur around the base flow streamlines but over increasingly smaller regions with more flow reversals. However, this paired with increased amplitudes, will promote a much greater hydrodynamic flow breakdown.

Decreasing κ confines the range of eigenvalues much nearer to the critical line and reduces the number of unstable circular frequencies. For $m = 0$, no instability is expected. Increasing to two and four is accompanied by successive increases in the number of unstable circular frequencies.

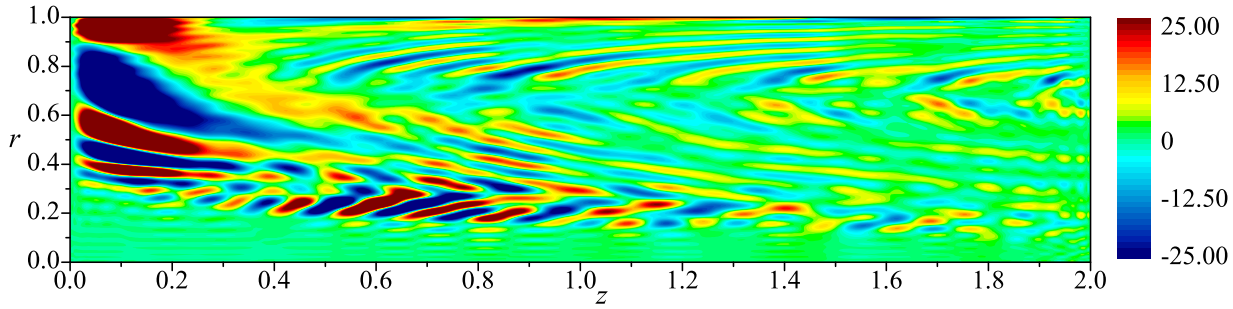
6.3.3 Evolution with Time

To understand the temporal evolution of the complex-lamellar flow breakdown, Fig. 6.13 shows the time evolution of the instantaneous axial velocity at $t = 10$, 20, 30, and 40 seconds. We have already observed that the magnitude of the perturbation is of the same order as the base flow and will destroy the base flow even without amplification. The axial velocity base flow profile can be seen for $t = 10$ sec but is unrecognizable for the other plots. Keep in mind that these plots are generated from the first (smallest) amplified eigenvalue at the specified parameters.

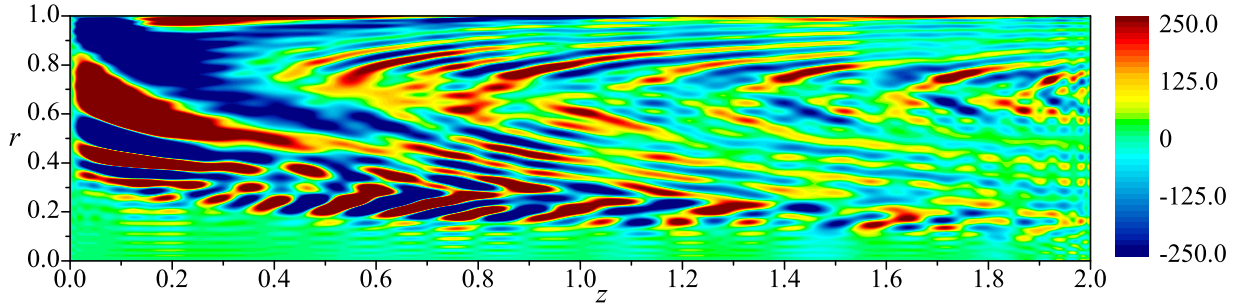
By turning our attention to the scale used in each figure, it becomes apparent that the amplified disturbances are so large that they exceed the range of applicability of the present, linear framework. From one perspective, we treat the fluctuation as a small amplitude perturbation. This stands in contradiction with the results. The other perspective is that oscillations remain predominantly linear and that nonlinear (higher order) contributions can be ignored. According to the latter perspective, these results remain acceptable. Either way, both views consider only exponential (linear) amplification as a result of the normal mode approach. Without nonlinear effects to impose a limit cycle amplitude, we can see amplification to the extremes shown here where in reality they would reach an asymptotic limit.



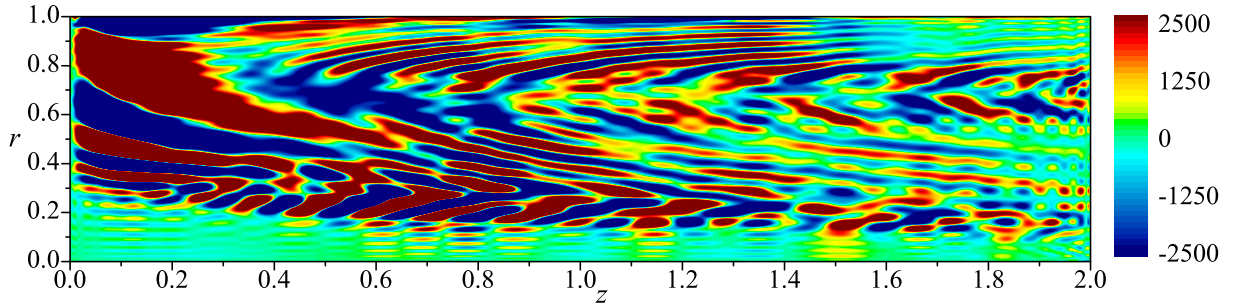
a) Biglobal $\tilde{U_z}$ at 10 seconds for the complex-lamellar model



b) Biglobal $\tilde{U_z}$ at 20 seconds for the complex-lamellar model



c) Biglobal $\tilde{U_z}$ at 30 seconds for the complex-lamellar model



d) Biglobal $\tilde{U_z}$ at 40 seconds for the complex-lamellar model

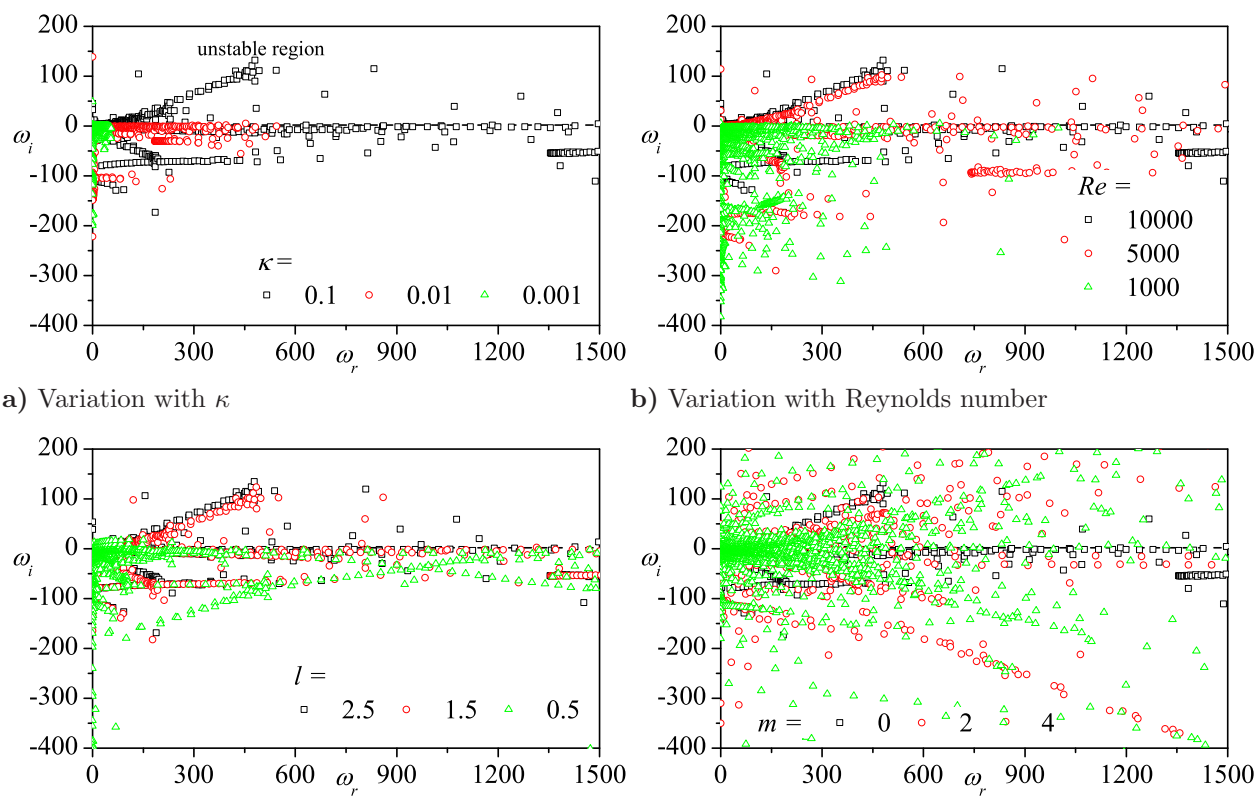
Figure 6.13: Temporal evolutions of the instantaneous axial velocity for the first unstable eigenvalue, $\omega = 0.2178 + 0.2940i$, and eigensolution of the complex-lamellar bidirectional vortex with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$.

6.4 The Linear Beltramian Bidirectional Vortex

The same parametric study is now applied to the linear Beltramian vortex. Given the notable difference in the vorticity profile between the Beltramain vortex base flows and the complex-lamellar due to the inclusion of axial dependence in the tangential velocity, we can expect slightly different behavior regarding hydrodynamic breakdown.

6.4.1 Axisymmetric Spectrum

Figure 6.14 illustrates a parametric study of several key parameters. Similar to the complex lamellar results, variations with respect to κ appear to be small when compared to other



a) Variation with κ

b) Variation with Reynolds number

c) Variation with aspect ratio

d) Multiple mantles

Figure 6.14: Axisymmetric parametric study for several input parameters. Here $q = 0$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

factors. Our results produce straight lines emanating from the origin with most eigenvalues falling near or below the critical line. The orientation of these straight line structures compares nicely with the previous model but extend further into the high frequency domain, especially for large κ. Changing the Reynolds number introduces a wider scattering of the spectrum below the critical line, as shown in Fig. 6.14b. Therein, the specra are given at five successive values of the Reynolds number ranging from 100 to 10000. Again, changing the aspect ratio brings about insubstantial changes in the amplification, but nearly no change in circular frequency for many of the eigenvalues. Here, the aspect ratio of one more closely relates to aspect ratios larger than one where the opposite was true for the complex-lamellar results. Finally, multiple mantle spectral results appear to be quite scattered. Coherent line-like structures still appear but are widely stretched over the frequency domain compared to the complex-lamellar model, except for the $\kappa = 0.01$ case. This property was seen in the LNP parametric study as well.

The associated waveforms with the axisymmetric results are identically (or very nearly zero) for all vector directions. As such, we must consider higher mode numbers to extract non-vanishing stability results.

6.4.2 Asymmetric Spectrum

Variations with respect to κ are shown in Fig. 6.15. Being analogous to the complex-lamellar case, the linear Beltramian spectrum is most densely populated near the origin. Eigenvectors for larger values of κ persist further into the high frequency domain and form interesting spectral structures around the abscissa. There is some evidence that smaller values of κ may be more stable than their larger counterparts. This suggests that increasing swirl has a stabilizing effect. Changing the Reynolds number in Fig. 6.16a leads to a greater degree of spectral disparity among the test cases. First, we see that smaller Reynolds numbers shift the spectrum to a lower frequency domain. Conversely, high frequency oscillations

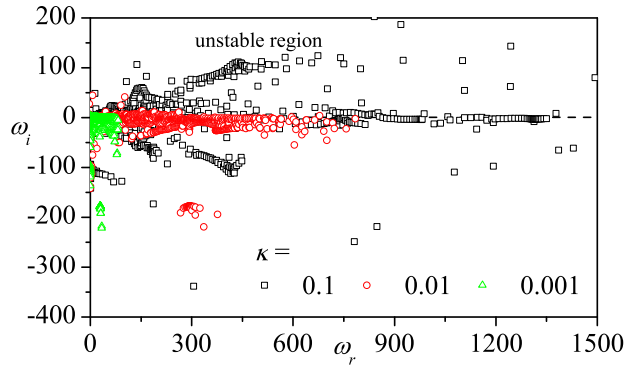
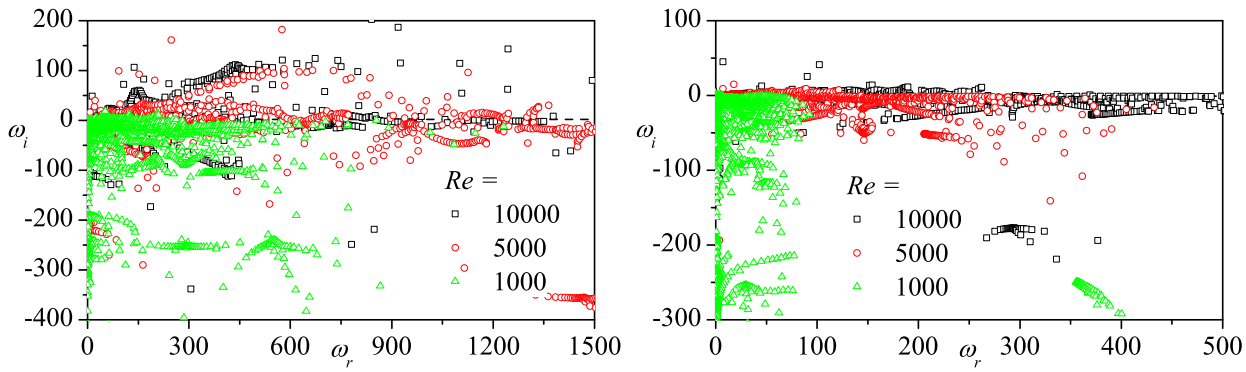


Figure 6.15: Asymmetric variation with κ. Here $q = 1$, $l = 2$, $Re = 10,000$.

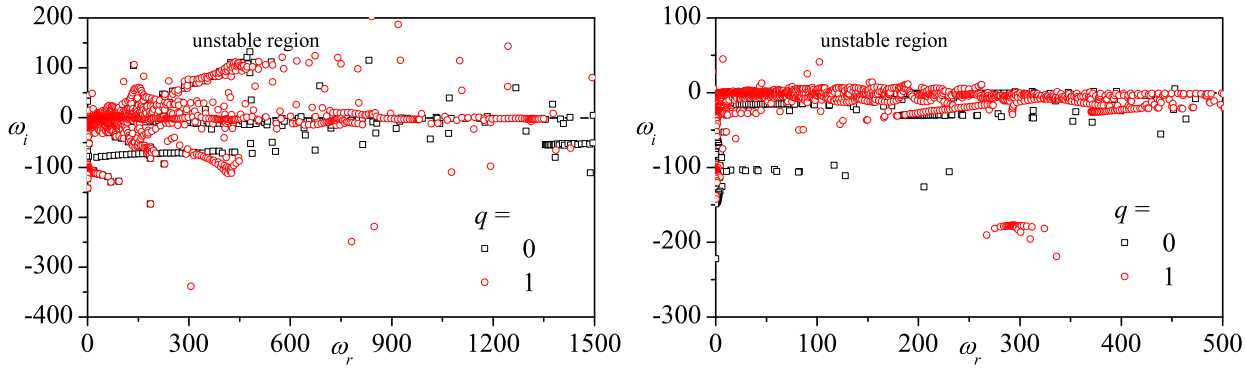


a) Variation with Reynolds number, $\kappa = 0.1$ **b)** Variation with Reynolds number, $\kappa = 0.01$

Figure 6.16: Asymmetric variation with Reynolds number. Here $q = 1$, $l = 2$, and $Re = 10,000$.

appear for larger values of Re. Furthermore, it can be seen that higher Reynolds numbers will induce higher amplitude instabilities. The corresponding study for $\kappa = 0.01$ is shown in Fig. 6.16b. Similar trends found in the complex-lamellar solutions are also seen here. Decreasing κ reduces the number of unstable eigenvalues significantly. Potentially unstable circular frequencies reside near $\omega_r = 100, 150,$ and 185 for $Re = 10000$. The other two cases of $Re = \{5000, 1000\}$ have no expected unstable frequencies. Also, as the Reynolds number decreases, the spectrum coalesces closer to the origin.

Again we see a substantial overlap of the spectra associated with higher mode numbers in Fig. 6.17. The axisymmetric spectrum ($q = 0$) is mostly confined to the lower



a) Variation with tangential mode number, $\kappa = 0.1$ **b)** Variation with tangential mode number, $\kappa = 0.01$

Figure 6.17: Spectral results for higher mode numbers. Here $l = 2$ and $Re = 10,000$.

frequency domain and is largely damped or slightly undamped in this region. It possesses higher frequency undamped eigenvalues, but these carry larger error related to the spectral resolution and may be misleading. The first asymmetric mode produces the most clearly defined spectral structures as well as a higher overall amplification than the remaining test cases. The higher transverse modes can be seen to exhibit similar spectral patterns as their predecessors with comparable moduli. The smaller value of κ confines the spectrum to a narrow band near the critical line for both tangential mode numbers considered and increases the overall stability.

Waveforms for the first amplified eigenvalue are shown in Fig. 6.18. These solutions bear familiar characteristics to that of the complex-lamellar solution. In particular, the regions and amplitudes of highest oscillation coincide for the two solutions. Again, the oscillations appear to concentrate along the streamlines of the base flow as noted by the solid lines; here too, the pressure fluctuation seems to have no appreciable effect. Furthermore, the funnel shaped contour of the inner vortex, marked by a dashed line, remains nearly undisturbed.

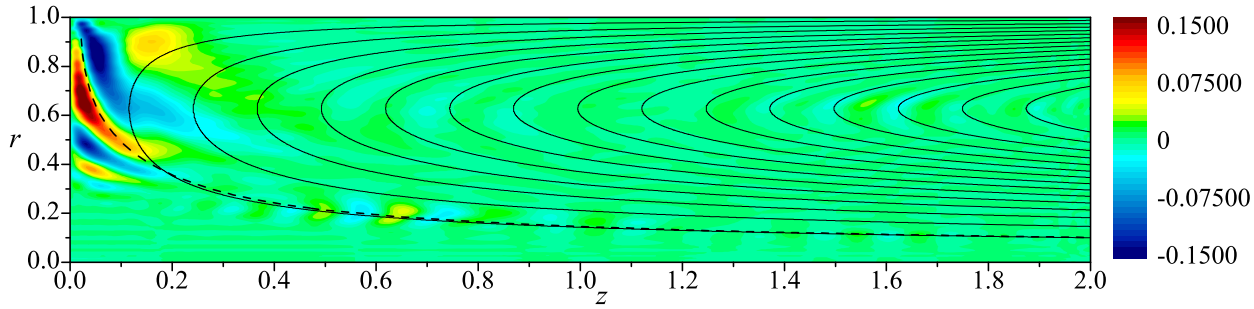
Unlike the previous results, the natural amplitude of these oscillations is not of the same order as the base flow for the axial velocity. Therefore, the higher magnitude instantaneous velocity is less dominated by the hydrodynamic oscillation. The radial component does,

however, appear at the same order as the base flow radial velocity, especially near the headwall (Fig. 6.18a).

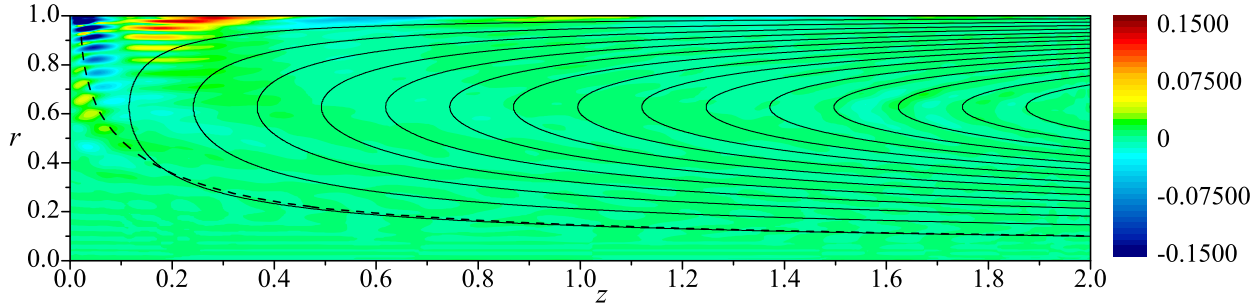
Representative contour plots for $\kappa = 0.01$ are shown in Fig. 6.19. Unlike the spectral results, the wave form does not exhibit the same similarities between the complex-lamellar and linear Beltramian figures. Here, oscillations occur within the chamber volume rather than only along the sidewall. Also, these solutions indicate spatial instability in the streamwise direction. This is not the case for larger κ nor the complex-lamellar solution. We can conclude that, once again, the greater influence of the tangential base flow velocity disallows the flow to oscillate around the base flow streamlines. It may be fruitful to note that the oscillations occur within the axially variant portion of the tangential velocity component. This region is rotational and could explain why volumetric oscillations were unable to form in the irrotational complex-lamellar solution. The pressure oscillations are small and are confined to the centerline and slowly oscillates in the axial direction.

Changing the Aspect Ratio and Multiple Mantles

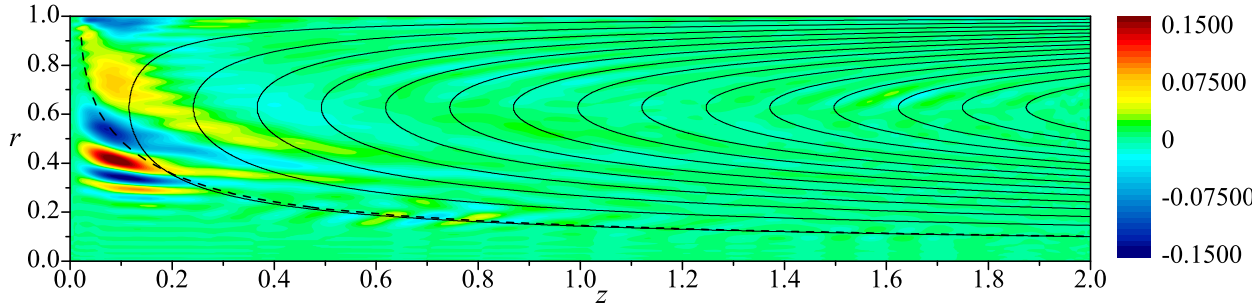
The effect of varying aspect ratio remains an interesting parametric study. Figure 6.20 shows a less amplified overall spectrum for an aspect ratio of 0.5. The spectral character overlaps closely for the remaining cases. The waveforms in Fig. 6.22 show similarities with the complex-lamellar results in that the highest amplitude oscillations occur near the headwall and appear to be drawn further downstream with larger l. Reducing κ improves the stability by reducing the number of unstable eigenvalue. Careful consideration suggests that unstable modes are possible near $\omega_r = 100, 130, 185$ for $l = 2.5$ although few sporadic unstable modes appear for the other two cases. Interestingly, the number of unstable eigenvalues reduces from $l = 2.5$ to 1.5 and then increases again for $l = 0.5$ This reinforces the observation that short chambers ($l < 1$) may behave differently from longer chambers.



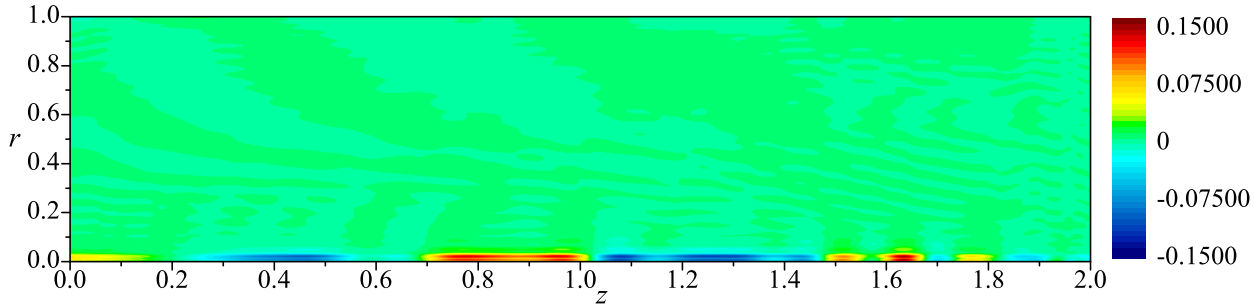
a) Biglobal radial velocity wave for the linear Beltramian model



b) Biglobal tangential velocity wave for the linear Beltramian model

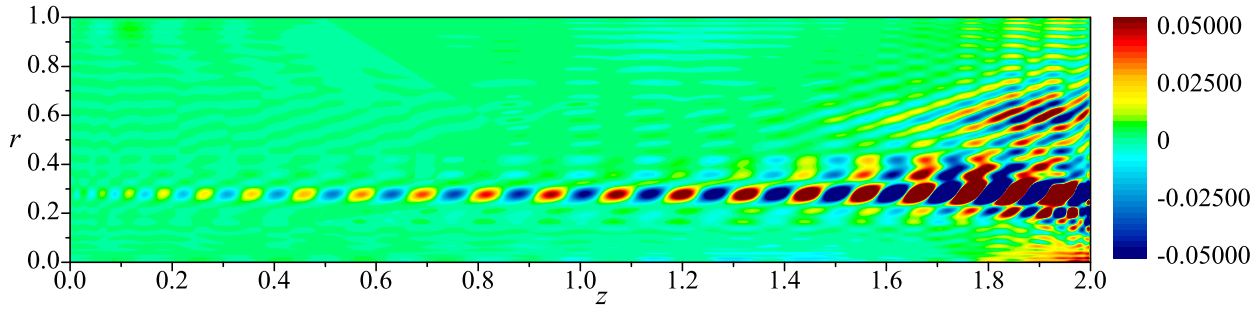


c) Biglobal axial velocity wave for the linear Beltramian model

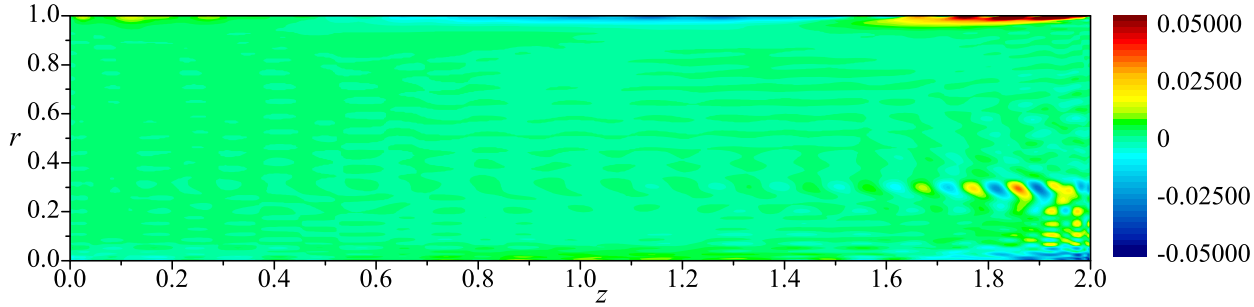


d) Biglobal pressure wave for the linear Beltramian model

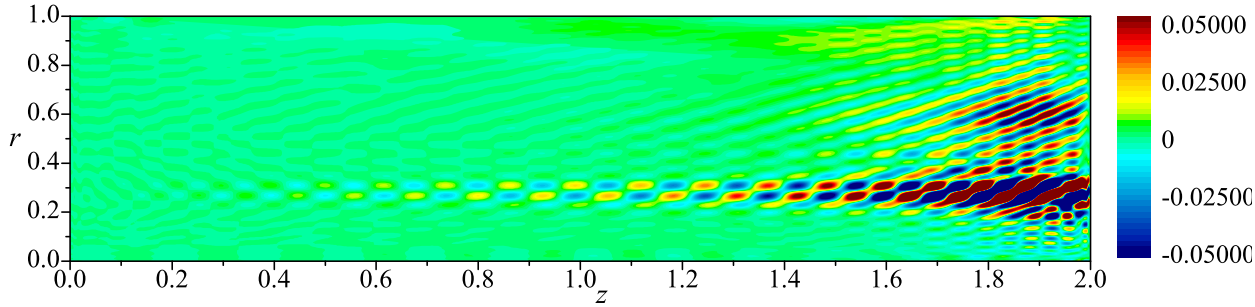
Figure 6.18: Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.2312 + 0.1096i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. See Table 6.2 for error values.



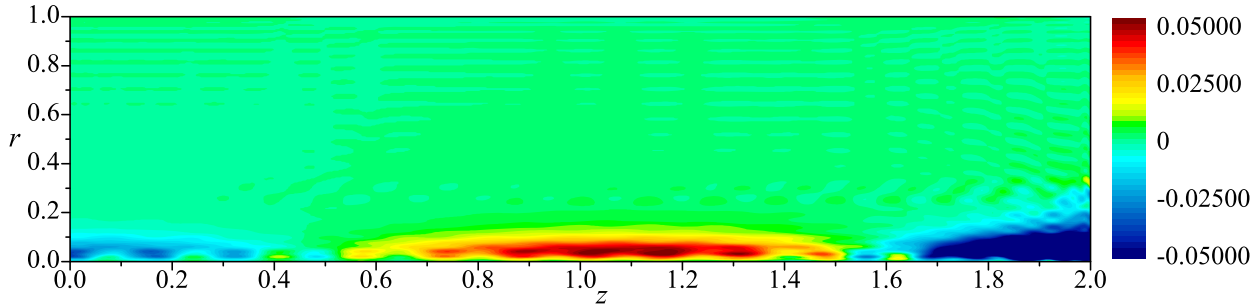
a) Biglobal radial velocity wave for the linear Beltramian model



b) Biglobal tangential velocity wave for the linear Beltramian model

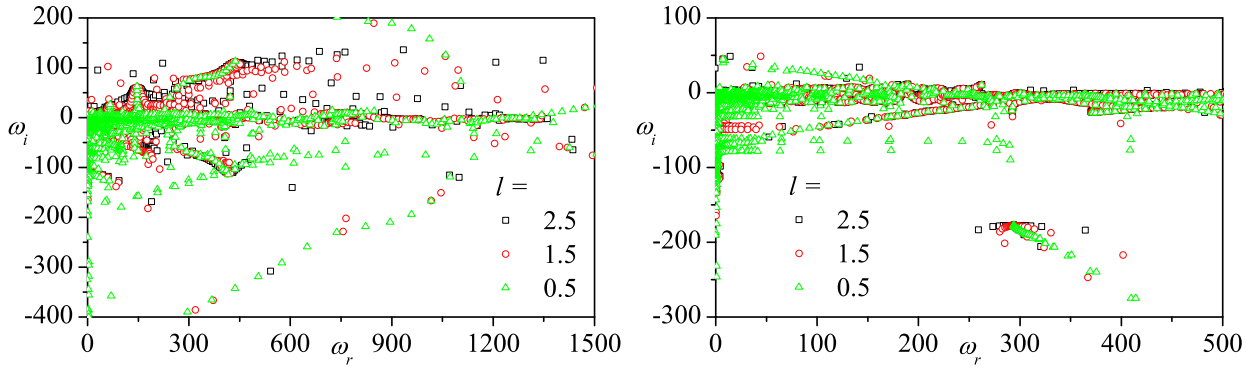


c) Biglobal axial velocity wave for the linear Beltramian model



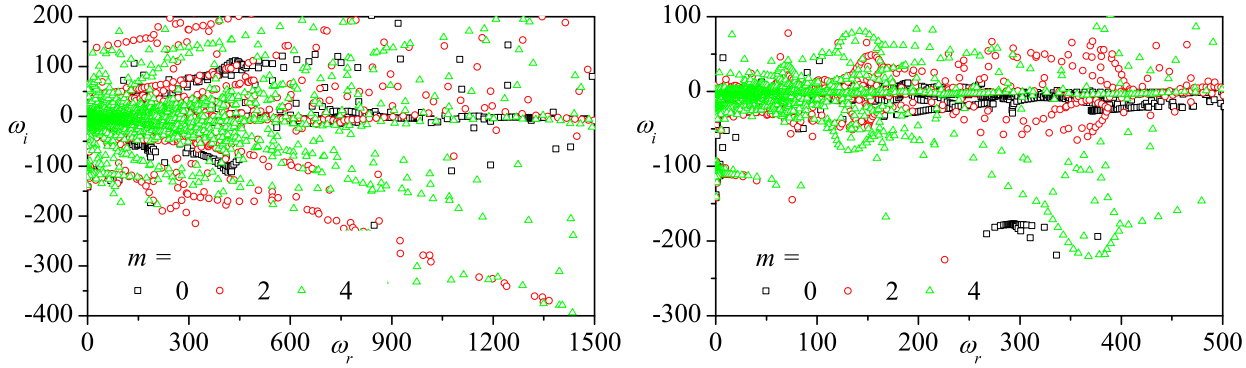
d) Biglobal pressure wave for the linear Beltramian model

Figure 6.19: Asymmetric eigensolutions for the unstable eigenvalue, $0.8992 + 0.1617i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.01$.



a) Variation with aspect ratio, $\kappa = 0.1$ **b)** Variation with aspect ratio, $\kappa = 0.01$

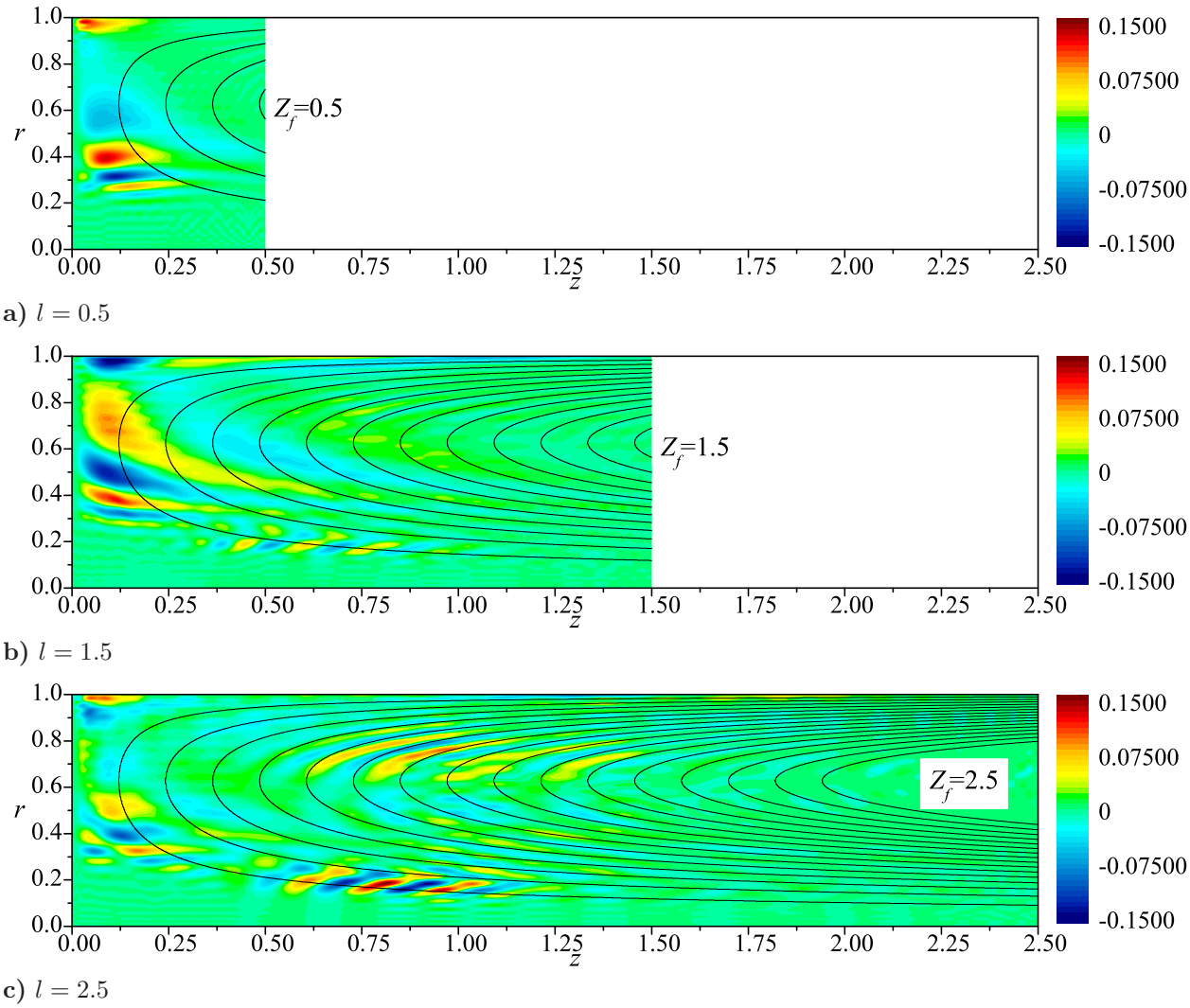
Figure 6.20: Spectral results for several values of the aspect ratio. Here $q = 1$ and $Re = 10,000$.



a) Variation with multiple mantles, $\kappa = 0.1$ **b)** Variation with multiple mantles, $\kappa = 0.01$

Figure 6.21: Spectral results for several values multidirectional flow. Here $q = 1$, $l = 2$, and $Re = 10,000$.

The multiple mantle spectral results in Fig. 6.21 and the waveforms in Fig. 6.23 show an increase in scatter compared to the complex-lamellar solutions. The largest concentration of eigenvalues appears near the centerline. Spectral structures are present but much more spread out. Although not conclusive from Fig. 6.21, increasing the number of flow reversals seems to increase the amplification of the undamped frequencies. This is consistent with the behavior displayed by the previous flow models. An expected increase in disorganization of the hydrodynamic wave accompanies successive increases in flow reversals, despite the small initial amplitudes captured here relative to the complex-lamellar model. Multi-directional



a) $l = 0.5$

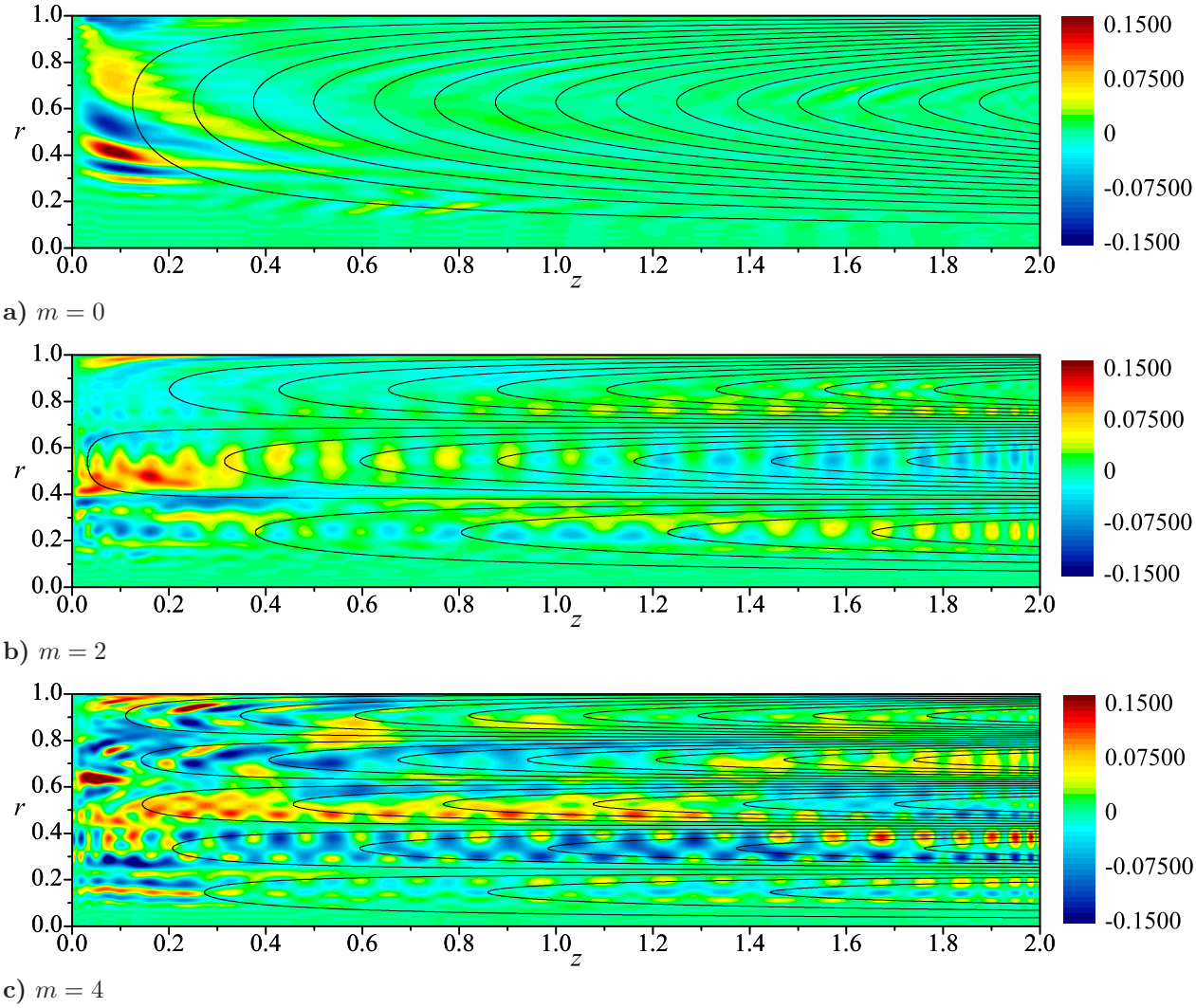
b) $l = 1.5$

c) $l = 2.5$

Figure 6.22: Axial waveform of the first unstable eigenvalue of the linear Beltramian bidirectional vortex for different aspect ratios with $q = 1$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$.

flow exhibits regular oscillations along the streamwise direction. This is not evident in the bidirectional results.

As expected, smaller κ reduces the number of unstable eigenvalues. While all cases of m contain unstable eigenvalues, the number of dangerous circular frequencies increases with increasing flow reversals, as does the expected growth rates.



a) $m = 0$

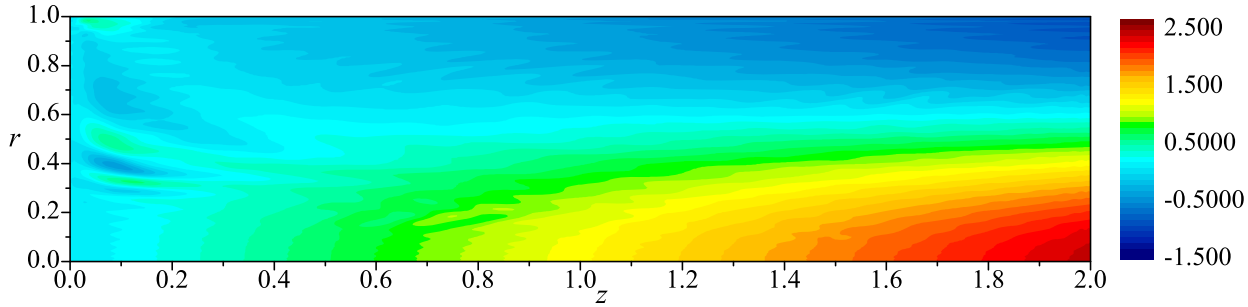
b) $m = 2$

c) $m = 4$

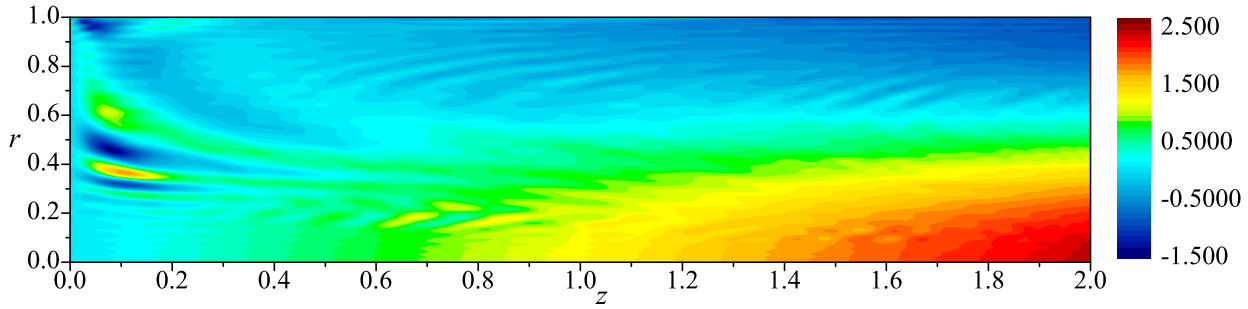
Figure 6.23: Axial waveform of the first unstable eigenvalue of the linear Beltramian bidirectional vortex for multidirectional flow with $q = 1$, $Re = 10,000$, and $\kappa = 0.1$.

6.4.3 Evolution with Time

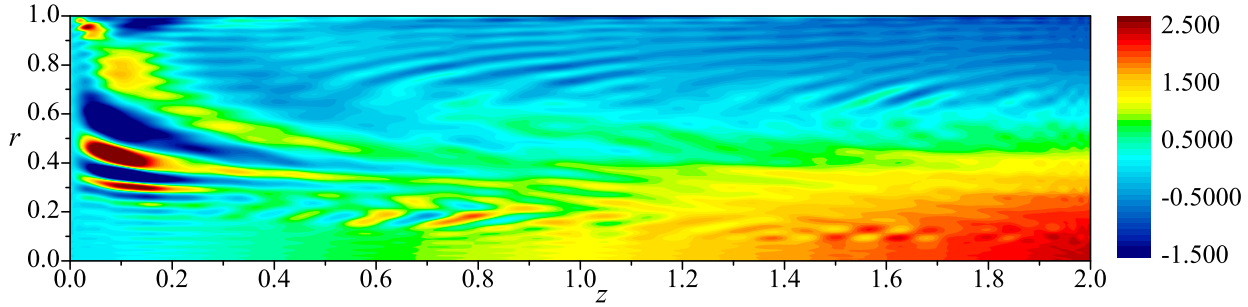
From Fig. 6.24, we see the steady degradation of the instantaneous axial velocity with time. Unlike the complex-lamellar solution, we see a much slower transition to turbulence here. In fact, after 10sec, there is only a minimal appearance of oscillations while at 30sec, the axial base flow is still a predominant component of the instantaneous velocity.



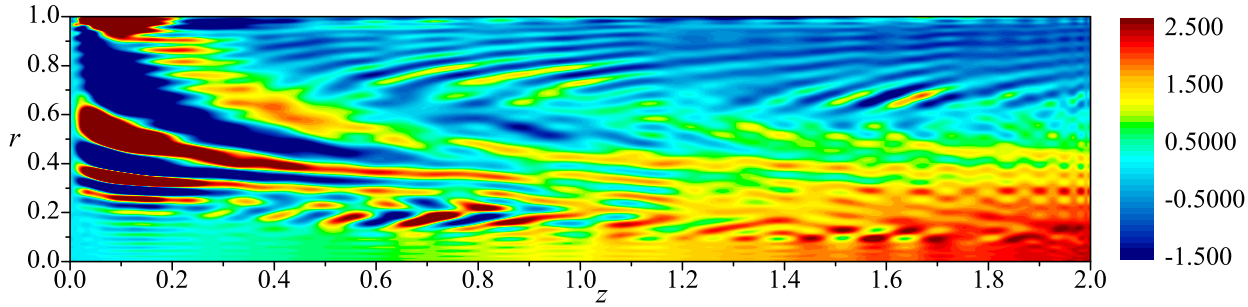
a) Biglobal $\tilde{U_z}$ at 10 seconds for the linear Beltramian model



b) Biglobal $\tilde{U_z}$ at 20 seconds for the linear Beltramian model



c) Biglobal $\tilde{U_z}$ at 30 seconds for the linear Beltramian model



d) Biglobal $\tilde{U_z}$ at 40 seconds for the linear Beltramian model

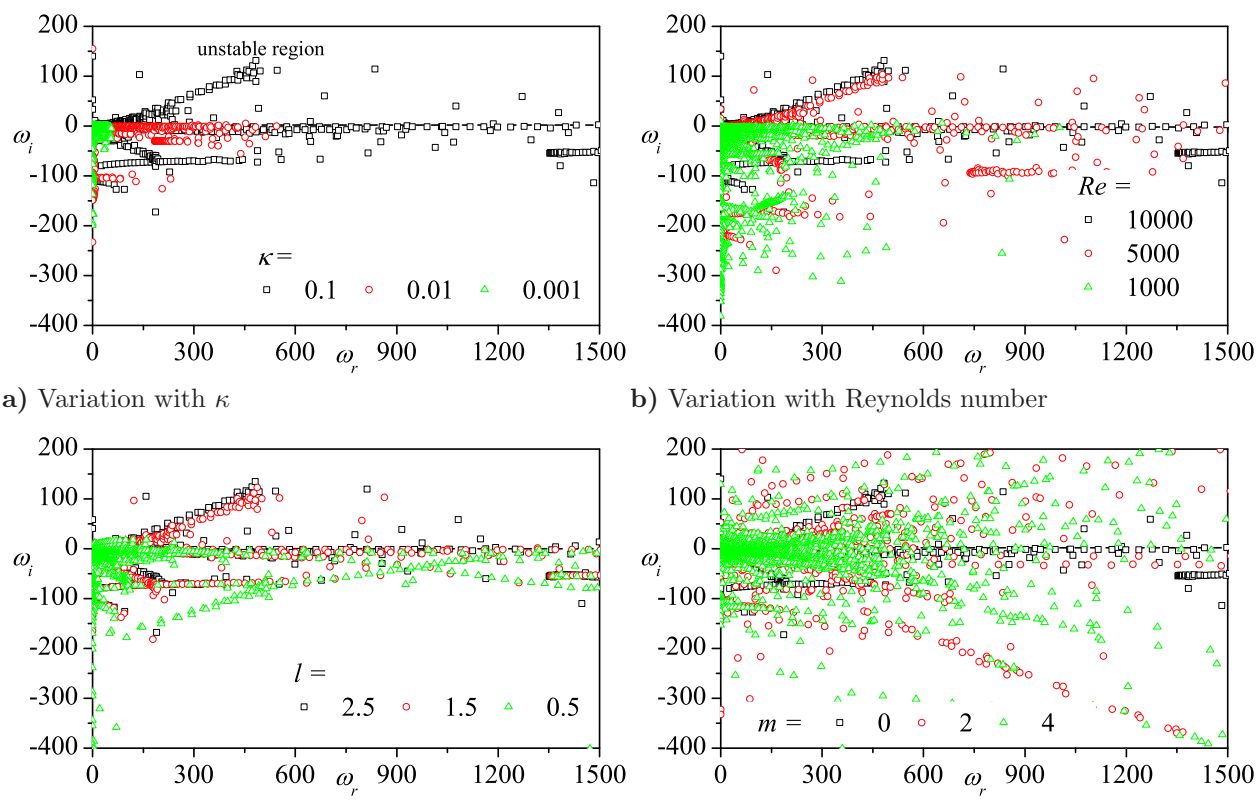
Figure 6.24: Temporal evolutions of the instantaneous axial velocity for the first unstable eigenvalue, $\omega = 0.2312 + 0.1096i$, and eigensolution of the linear Beltramian bidirectional vortex with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$.

6.5 The Harmonic Beltramian Bidirectional Vortex

Spectral results for the harmonic Beltramian model mirror the linear model to a high degree. While similar comparisons were found for both Beltramian models using the LNP approach these similarities become even more exacting with the biglobal approach. There are, however, differences in the waveforms to be reported.

6.5.1 Axisymmetric Spectrum

The parametric study for the harmonic Beltramian base flow in Fig. 6.25 presents nearly identical results to those reported for the linear Beltramian model in Fig. 6.14. As before,



a) Variation with κ **b)** Variation with Reynolds number

c) Variation with aspect ratio **d)** Multiple mantles

Figure 6.25: Axisymmetric parametric study for several input parameters. Here $q = 0$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$ unless varied on the graph.

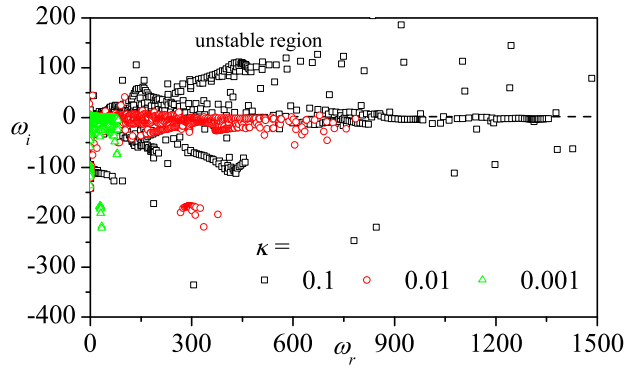


Figure 6.26: Asymmetric variation with κ. Here $q = 1$, $l = 2$, and $Re = 10,000$.

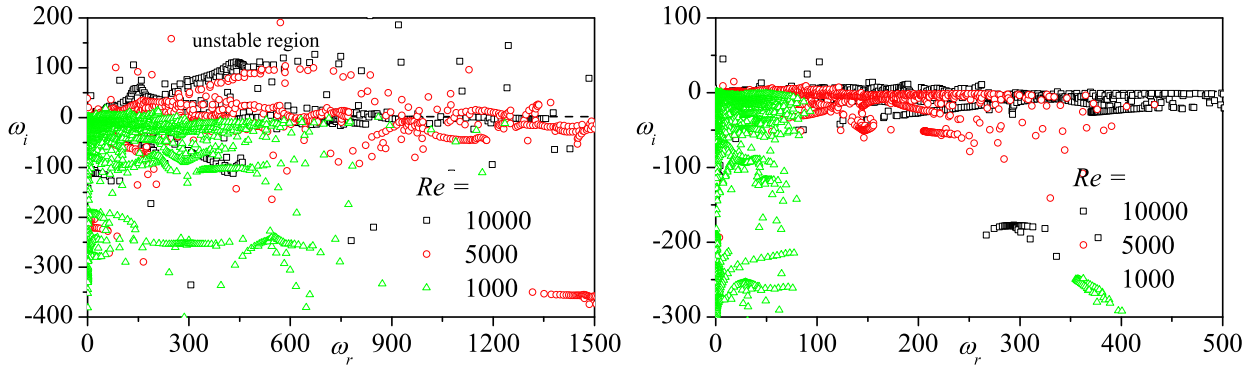
overlap of the spectral results accompanies variations in κ, with eigenvalues appearing at higher frequencies for large κ. The scatter of the overall spectrum with respect to Re is reminiscent of that in Fig. 6.14b. For both aspect ratio variations and multiple mantles, the results almost exactly coincide with those for the linear Beltramian case.

Consistent with all axisymmetric cases studied thus far, the waveforms associated with these spectral results are identically zero or very nearly zero. Regardless of how far above the critical line an amplified eigenvalue falls, it causes little to no hydrodynamic flow breakdown.

6.5.2 Asymmetric Spectrum

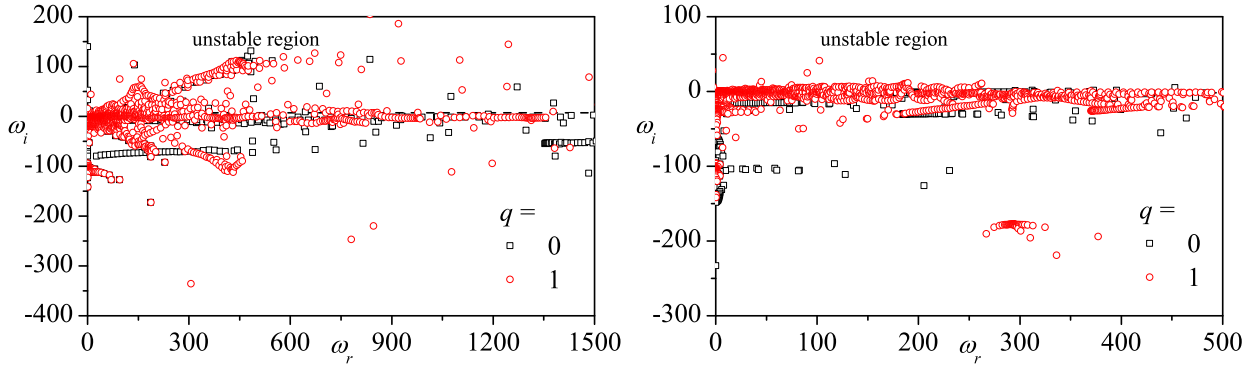
Following with the results of the linear Beltramian model, Fig. 6.26 shows variations in the spectrum with changes in κ. The effect of Reynolds number is evident in the large scatter of the spectrum for lower Re. Figure 6.27 suggests that lower Reynolds number solutions will tend to be more stable given that the majority of eigenvalues fall below the critical line of $\omega_i = 0$. The effect of reducing κ is consistent with the previous studies. The spectrum is more stable overall and does not contain high frequency eigenvalues.

Consistent results are found for mode numbers greater than zero in Fig. 6.28 with a slightly larger predicted amplitude for the $q = 1$ mode. This increase in the range of amplified modes at higher tangential mode numbers is moderate as the majority of eigenvalues recorded



a) Variation with Reynolds number, $\kappa = 0.1$ **b)** Variation with Reynolds number, $\kappa = 0.01$

Figure 6.27: Asymmetric variation with Reynolds number. Here $q = 1$, $l = 2$, and $Re = 10,000$.



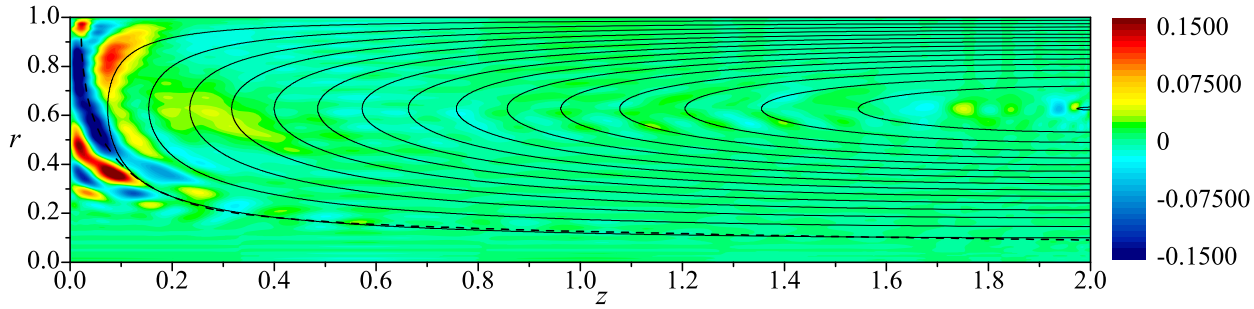
a) Variation with tangential mode number, $\kappa = 0.1$ **b)** Variation with tangential mode number, $\kappa = 0.01$

Figure 6.28: Spectral results for higher mode numbers. Here $l = 2$ and $Re = 10,000$.

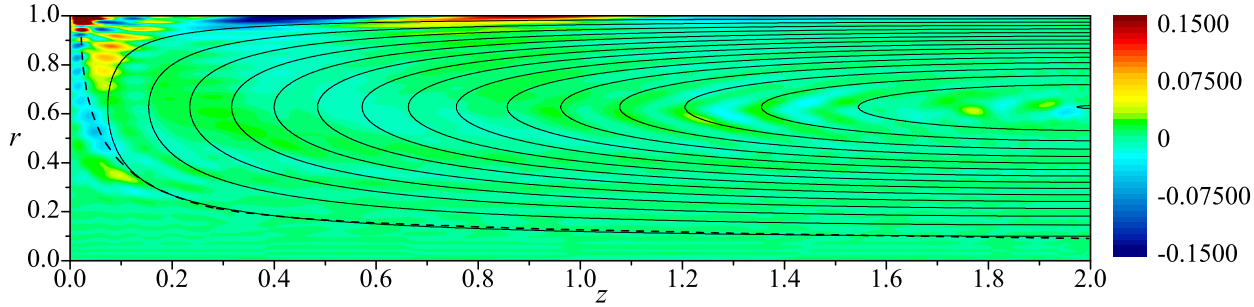
fall near or below the critical line. The smaller value of κ indicates an increase in overall stability.

Although the spectra are nearly identical between the two Beltramian solutions, the resulting waveforms are notably different (Fig. 6.29). The regions most affected by the hydrodynamic wave are significantly reduced and distributed slightly differently. However, the appearance of the base flow streamlines is still apparent throughout this analysis.

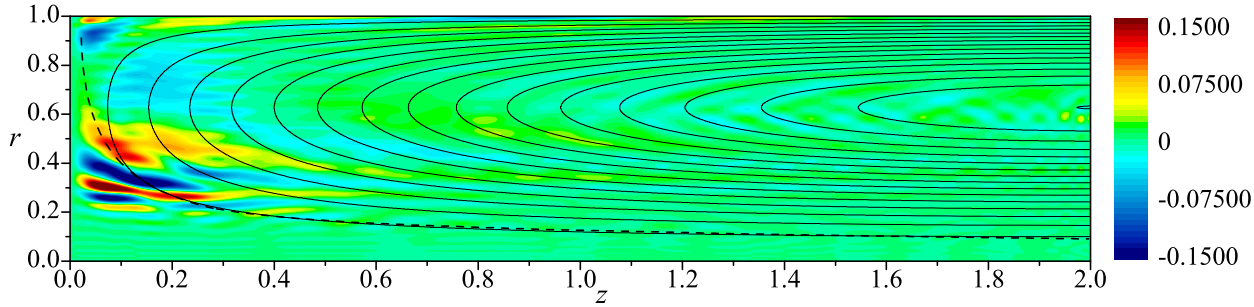
The relative size between the oscillations and the base flow is even smaller yet for this base flow. Although the difference is not significant, the resulting instantaneous velocity is even less characterized by the hydrodynamic solution than the previous examples. Coupled



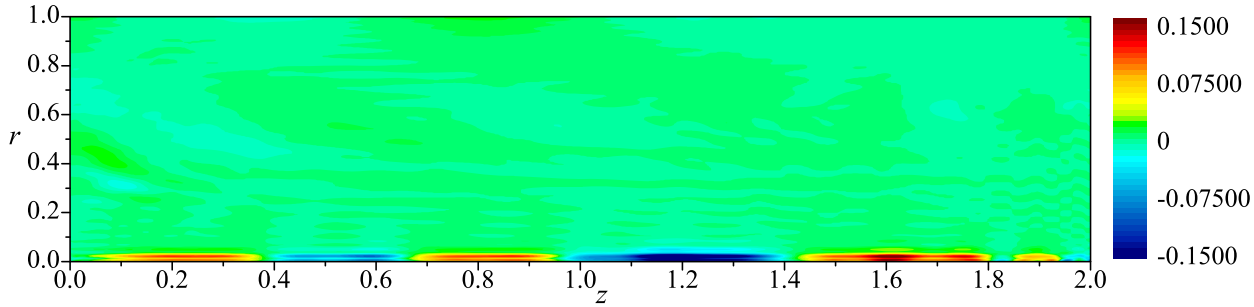
a) Biglobal radial velocity wave for the harmonic Beltramian model



b) Biglobal tangential velocity wave for the harmonic Beltramian model

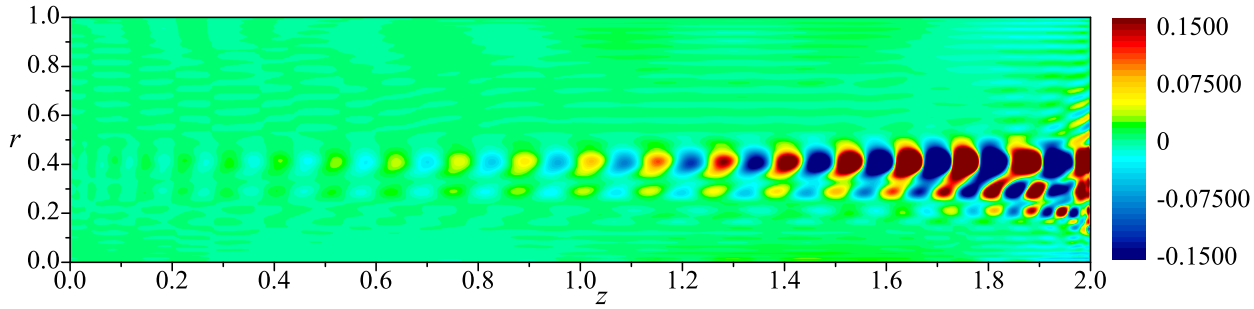


c) Biglobal axial velocity wave for the harmonic Beltramian model

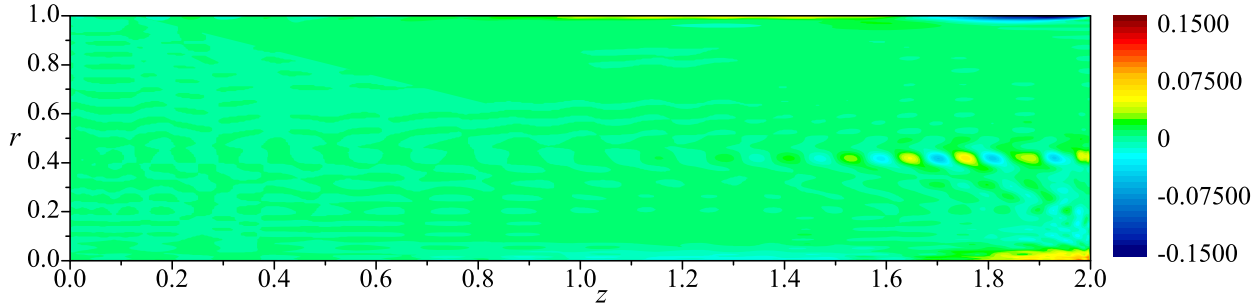


d) Biglobal pressure wave for the harmonic Beltramian model

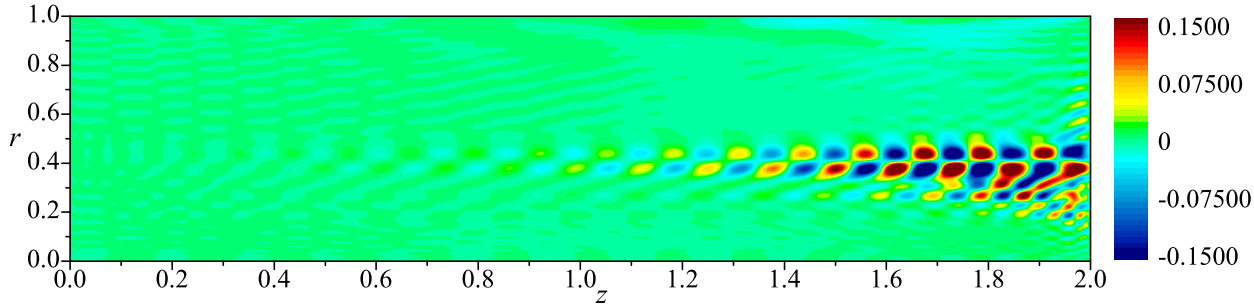
Figure 6.29: Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.0726 + 0.0549i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$. See Table 6.2 for error values.



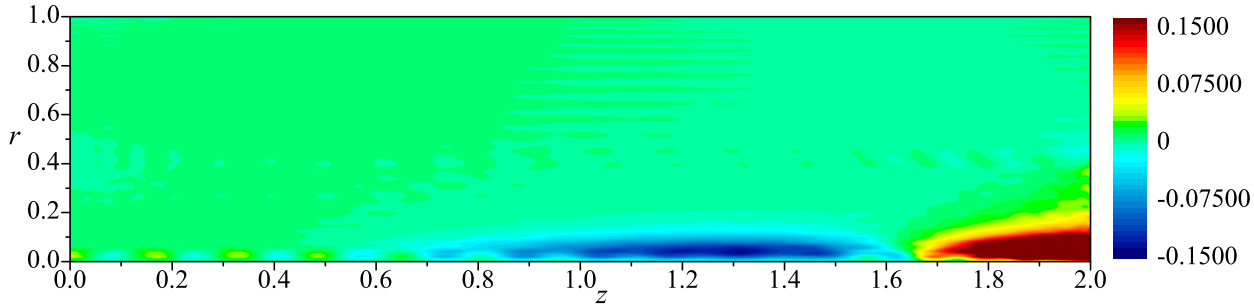
a) Biglobal radial velocity wave for the harmonic Beltramian model



b) Biglobal tangential velocity wave for the harmonic Beltramian model

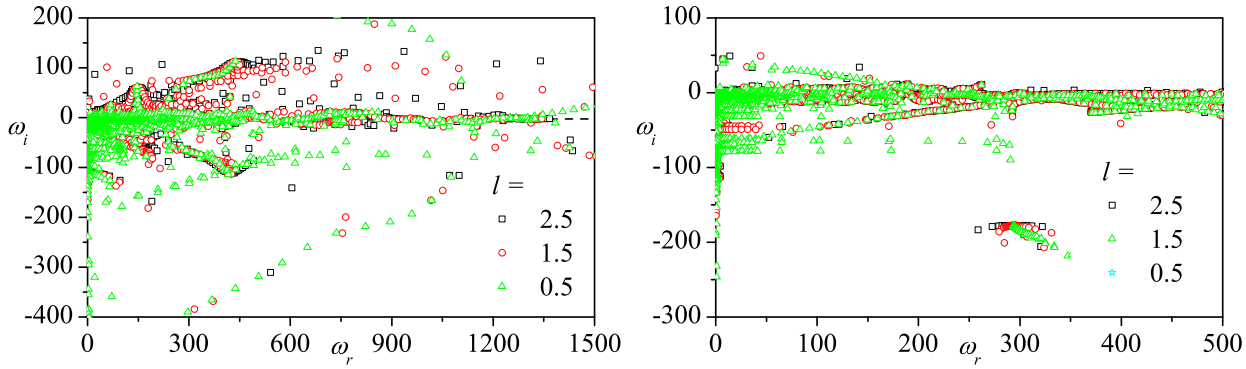


c) Biglobal axial velocity wave for the harmonic Beltramian model



d) Biglobal pressure wave for the harmonic Beltramian model

Figure 6.30: Asymmetric eigensolutions for the unstable eigenvalue, $\omega = 0.1734 + 0.1602i$, with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.01$.



a) Variation with aspect ratio, $\kappa = 0.1$ **b)** Variation with aspect ratio, $\kappa = 0.01$

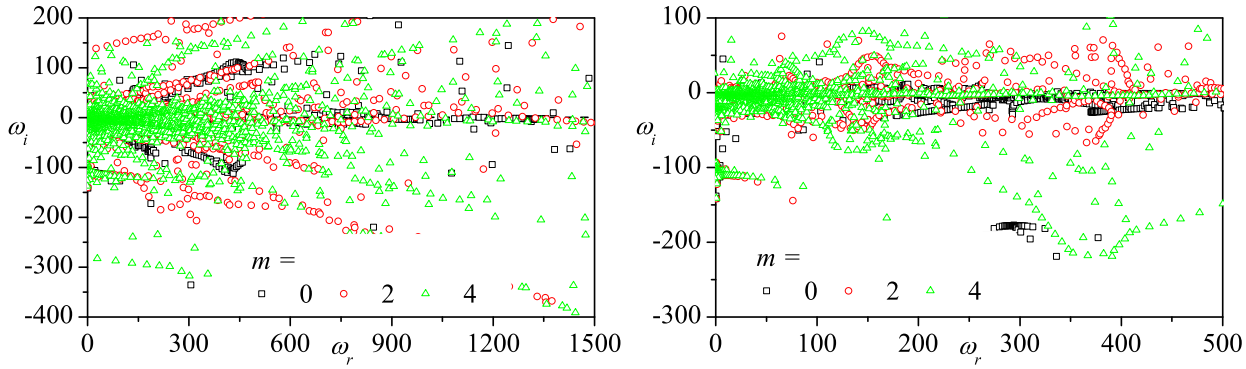
Figure 6.31: Spectral results for several values of the aspect ratio. Here $q = 1$ and $Re = 10,000$.

with the smaller regions of high amplitude oscillation, the overall flow can be assumed to be less prone to turbulent breakdown.

Contour plots involving a smaller value of κ in Fig. 6.30 show an obvious similarity to those in the linear Beltramian model. Specifically, oscillations appear away from the radial boundaries and increase in amplitude in the streamwise direction. Pressure oscillations are confined to the centerline and are characterized by a slow axial oscillation.

Changing the Aspect Ratio and Multiple Mantles

The spectrum shown in Fig. 6.31 considers varying aspect ratios. Since the harmonic Beltramian base flow is only defined up to l, this could be expected to cause a more significant effect on the spectral character than what actually results. This plot is very similar to the corresponding linear Beltramian plot. The eigenfunction contours in Fig. 6.33 do show a small difference between the two Beltramian solutions. We witness a large amplitude oscillation at the endwall for the $l = 0.5$ case which is not present in the previous examples. Also, the flow away from the wall is less oscillatory around the base flow streamlines than either the complex-lamellar or linear solutions. As for headwall oscillations, they are still drawn farther downstream with increased chamber length.



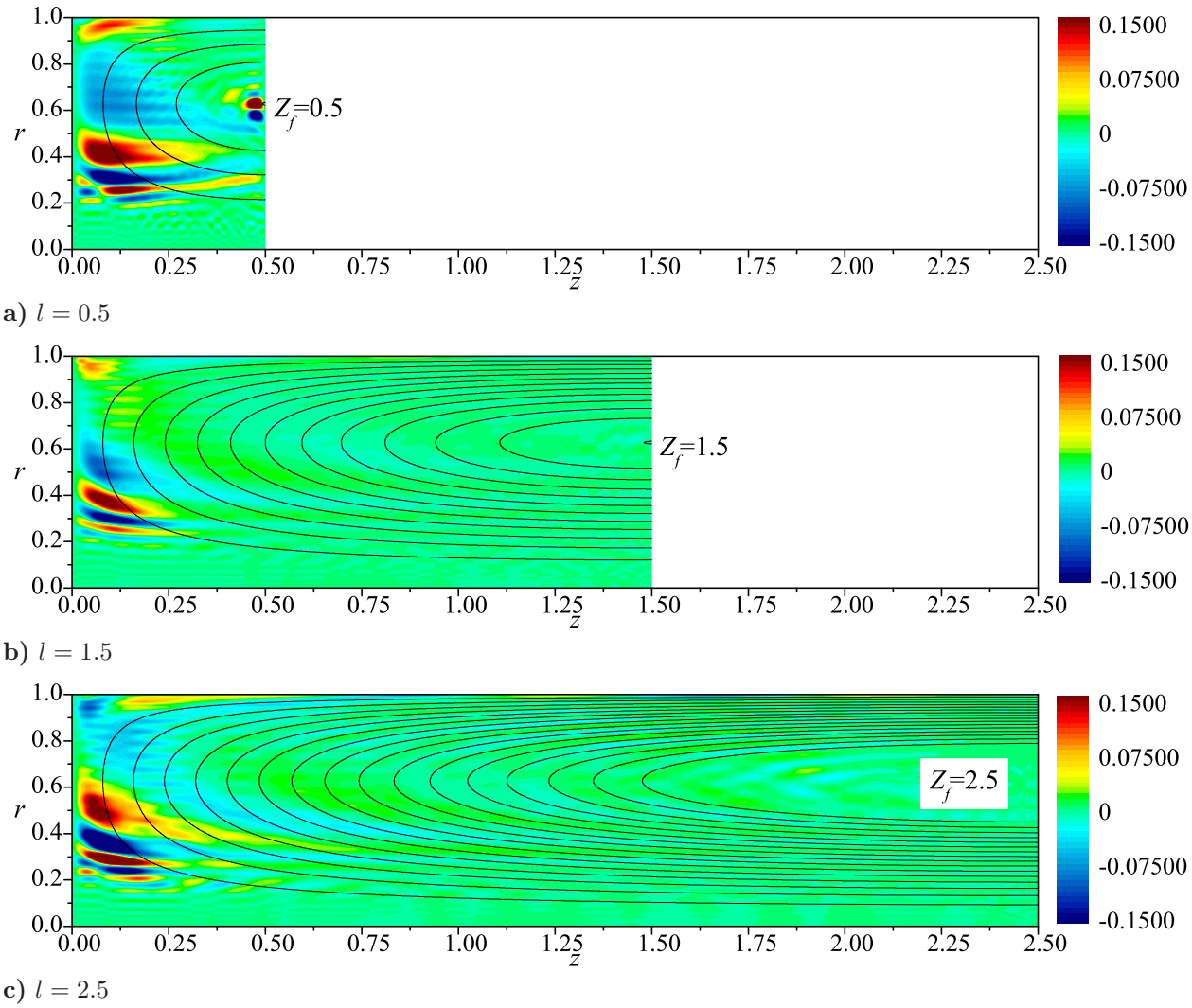
a) Multidirectional flow, $\kappa = 0.1$ **a)** Multidirectional flow, $\kappa = 0.01$

Figure 6.32: Spectral results for multidirectional flow. Here $q = 1$ and $Re = 10,000$.

Figure 6.32 gives the spectrum for multidirectional flow. Its apparent similarity to its linear Beltramian counterpart is somewhat lost in the eigenvector contours of Fig. 6.34. Although the contours match the linear Beltramian counterpart less closely than the corresponding spectrum, they still suggest that increased flow reversals will increase the effect of the hydrodynamic wave breakdown and lead to greater disruption of the base flow.

6.5.3 Evolution with Time

The time evolution plots in Fig. 6.35 are the best example yet of a slowly degrading flowfield. Even for $t = 30$ sec, hydrodynamic breakdown of the axial velocity does not degrade the base flow profile appreciably. Along with the small regions of high oscillation, the very low frequency amplified eigenvalue considered here will not be a major source of oscillation over short durations.



a) $l = 0.5$

b) $l = 1.5$

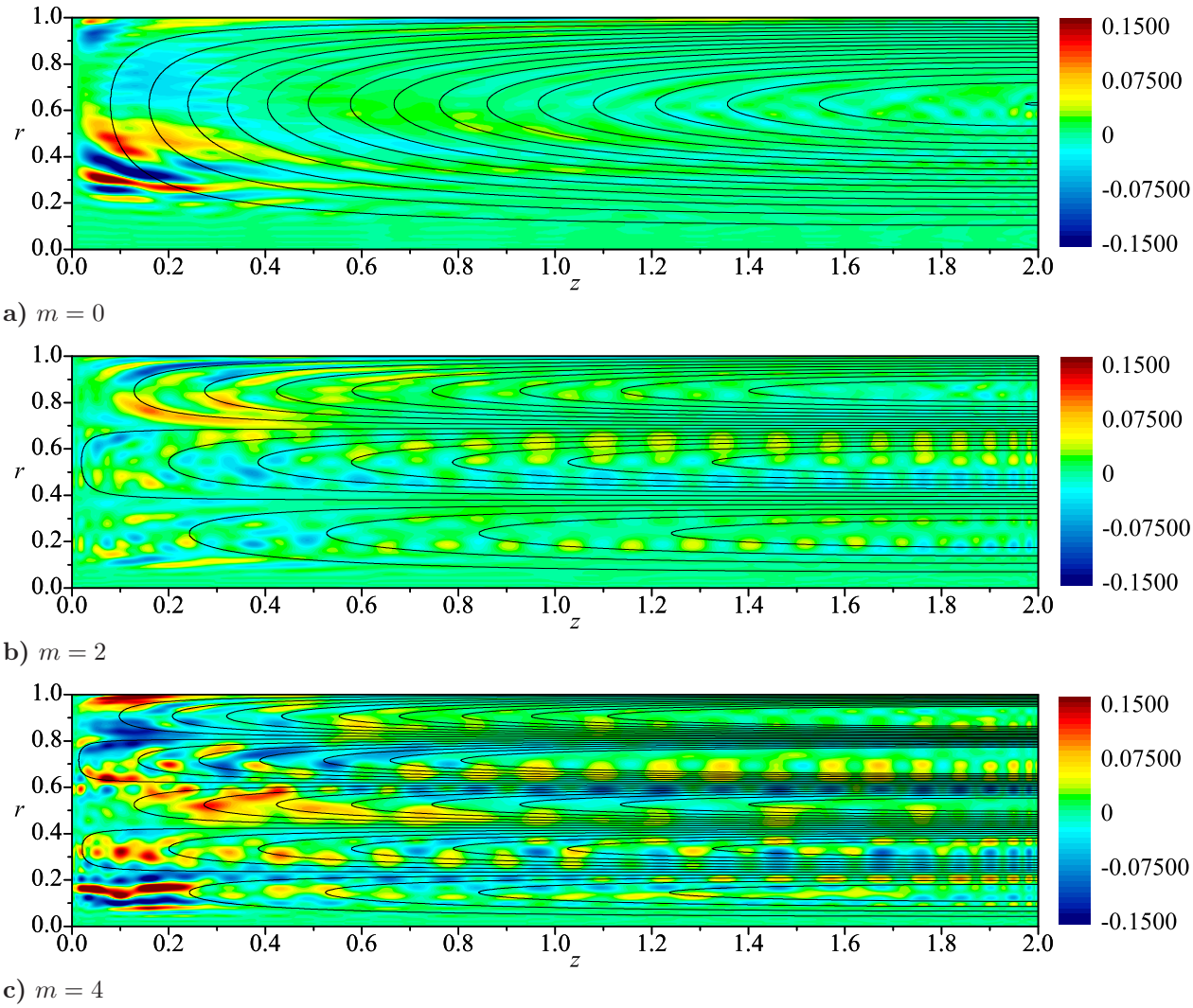
c) $l = 2.5$

Figure 6.33: Axial waveform of the first unstable eigenvalue of the harmonic Beltramian bidirectional vortex for different aspect ratios with $q = 1$, $l = 2$, $Re = 10,000$, and $\kappa = 0.1$.

6.6 Closing Remarks on the BG Approach

6.6.1 Considering the Waveforms

This chapter highlights results derived from biglobal stability analysis. To be slightly more complete, we continue our discussion here. The axisymmetric results are a clear example of how an amplified eigenvalue accompanied with a zero waveform does not contribute to



a) $m = 0$

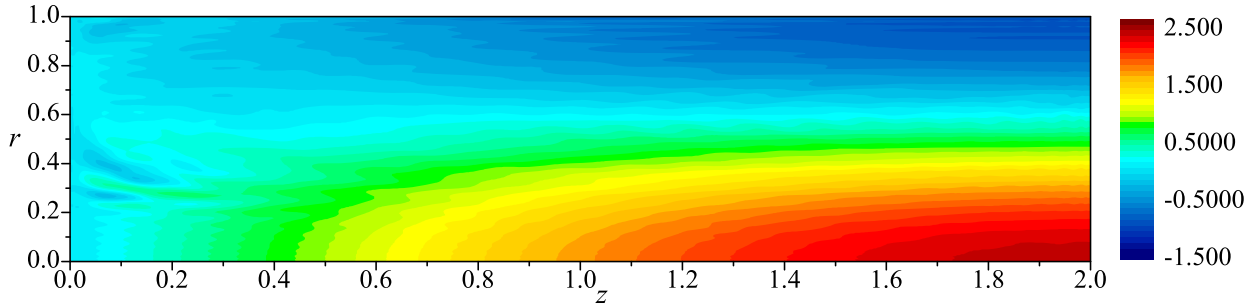
b) $m = 2$

c) $m = 4$

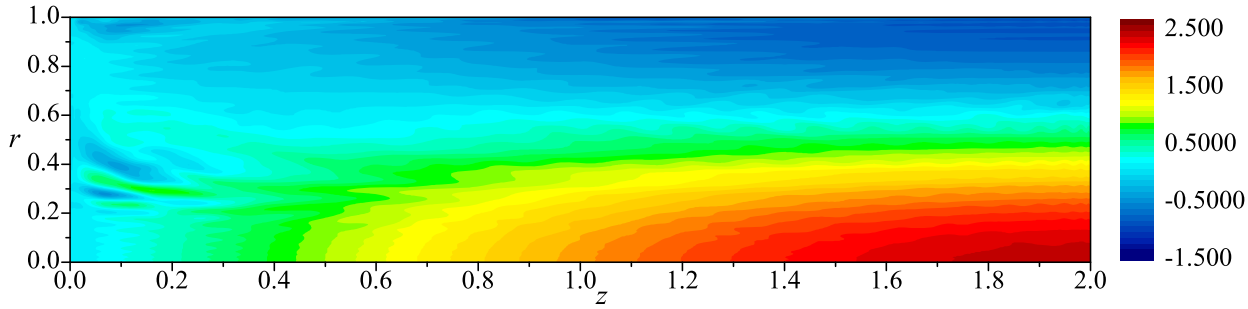
Figure 6.34: Axial waveform of the first unstable eigenvalue of the harmonic Beltramian bidirectional vortex for multidirectional flow with $q = 1$, $Re = 10,000$, and $\kappa = 0.1$.

hydrodynamic breakdown. It goes without saying that the overall stability is dependent on the total sum of all physical transverse and lateral disturbances. Thus it is important to consider the higher mode ($q > 0$) results.

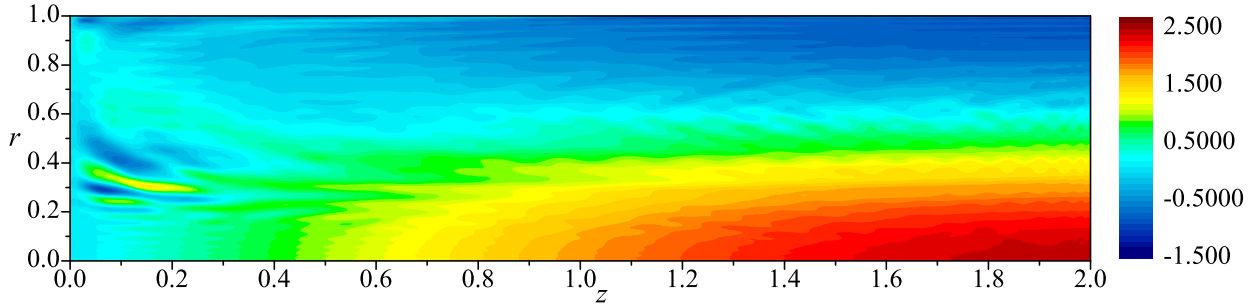
In this vein, it is important to see that while the eigenvalues control frequency and amplification, the original wave form is very important too. In fact, the waveforms shown above are for plausible configurations in propulsion but, outside of the expected domain, very



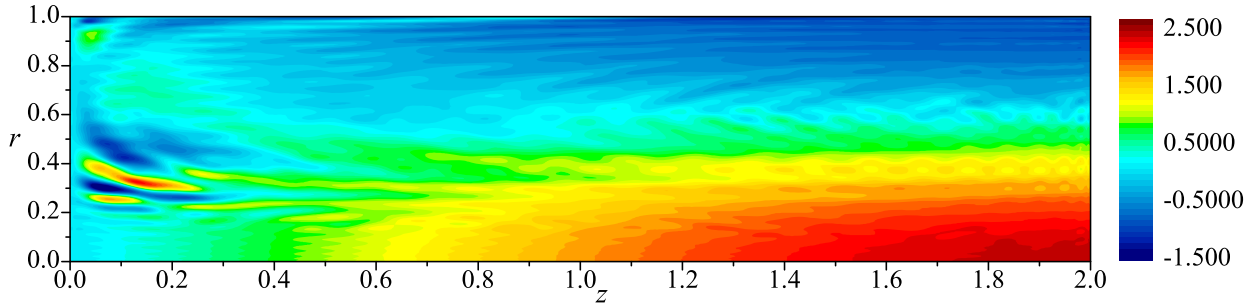
a) Biglobal $\tilde{U_z}$ at 10 seconds for the harmonic Beltramian model



b) Biglobal $\tilde{U_z}$ at 20 seconds for the harmonic Beltramian model



c) Biglobal $\tilde{U_z}$ at 30 seconds for the harmonic Beltramian model

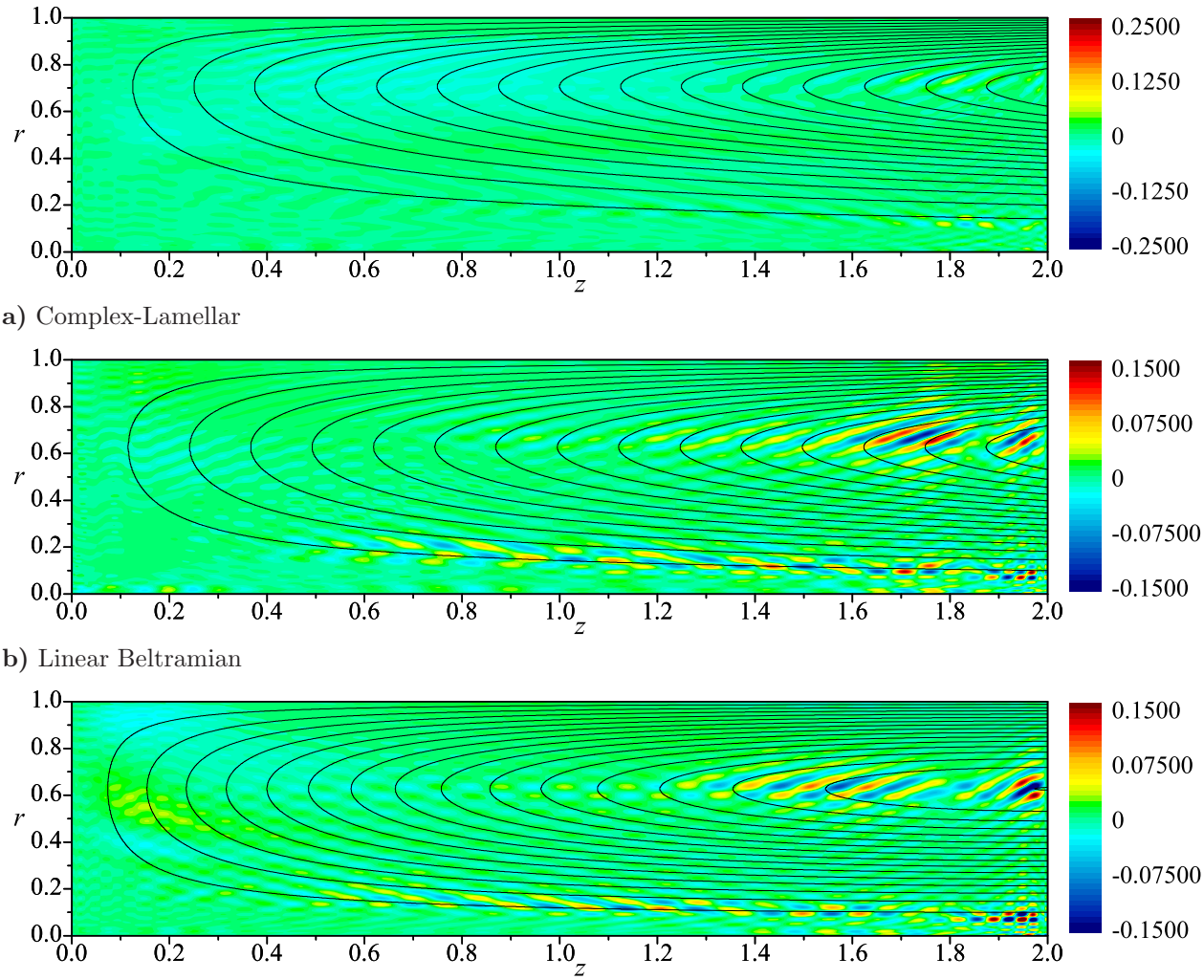


d) Biglobal $\tilde{U_z}$ at 40 seconds for the harmonic Beltramian model

Figure 6.35: Temporal evolutions of the instantaneous axial velocity for the first unstable eigenvalue, $\omega = 0.0726 + 0.0549i$, and eigensolution of the harmonic Beltramian bidirectional vortex with $N = 50$, $q = 1$, $Re = 10,000$, and $\kappa = 0.1$.

different results can be shown. One good example is the difference between the waveforms of a high and low Reynolds number flow. The variation of the spectrum is shown to be significant in the first place, but the character of the waveform is equally significant. From Fig. 6.36, we can compare a low Reynolds number solution for the linear Beltramian vortex to the high Re solution of the axial velocity in the previous contour plots. Immediate differences are at hand. First we do not see the region of high magnitude oscillations near the headwall. Small wavelength oscillations occur much further downstream and over much smaller regions. Furthermore, these oscillations produce a subtle grid-like effect in the contour plot. This type of waveform also appeared sporadically in the solutions by Chedevergne [123].

The spatial stability analysis by Abu-Irshaid, Majdalani, and Casalis [119] showed higher growth near the headwall. At the time, this result was difficult to justify given the downstream location of spatial growth in all previous studies. It was expected that since the inner region appears to be a truncated Taylor-Culick flow with the mantle replacing the transpiring wall, a similar spatial stability regime would develop. The biglobal waveform confirms that the region of largest spatial wave amplitudes are indeed near the headwall for large Reynolds number flows. Casalis and Vuillot [55] assert that flow patterns with severe curvature of their streamlines will tend to be more unstable than those with parallel flow. From this perspective, the region where the bidirectional vortex streamlines experience the most severe curvature appears at the headwall where the flow negotiates a 180 degree turn. Upon seeing this coincide with the region of largest amplitude waveforms, the one-dimensional results by Abu-Irshad should no longer be a surprise. We also note that downstream oscillations reach highest amplitudes at the top of the streamlines. These are also locations where the curvature of the mean flow streamlines is most severe.



a) Complex-Lamellar

b) Linear Beltramian

c) Harmonic Beltramian

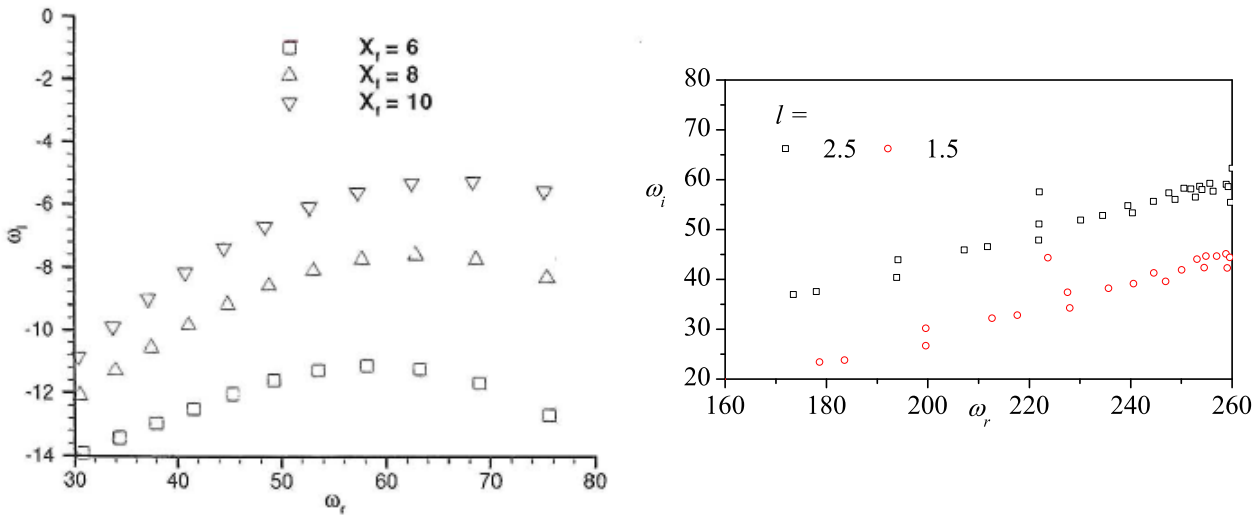
Figure 6.36: Axial wave contour for the first unstable eigenvalue $N = 50$, $q = 1$, $Re = 2,000$, and $\kappa = 0.1$.

6.6.2 Data Reduction

The dense scatter of eigenvalues, especially near the origin and neutral line of $\omega_i = 0$, makes stability predictions of the flowfield difficult. This will remain a difficult achievement for any analysis in which the streamfunction formulation, and therefore a significantly less dense spectrum, cannot be used. When considering the effect of hydrodynamic stability from a combustion instability framework where solutions are of the form $\exp(\alpha t)$, and α represents

the total growth rate, we are inclined to consider the overall spectral character. By simply averaging the imaginary component, we find that the spectrum has an overall negative growth rate for all cases considered. This is merely a qualitative means of discussing the results. While only one eigenvalue needs to be undamped to cause serious instabilities, the cumulative contributions of their imaginary components at a discrete circular frequencies makes up the real growth rate. For a densely populated spectrum where several unique circular frequencies overlap or nearly overlap, post processing of the spectral results can be performed to identify potentially dangerous circular frequencies. Specifically, the average growth rate for discrete intervals can be calculated to identify unstable circular frequencies. By considering all eigenvalues falling within a small interval, the densely populated spectra can be reduced to a more manageable set. While a level of uncertainty still exists, unstable frequencies can more precisely be identified. These calculations have been applied as a post-process computation for all spectral plots presented in this chapter. The results are tabulated in App. C.

When considering the example eigenvalue problems in Ch. 3, we acknowledged that the first $N/3$ eigenvalues were accurate for a single equation eigenvalue problem. This notion allows some data reduction by considering only the first $4N^2/3$ eigenvalues. This number is an extrapolated value given the increase from one equation to four and one coordinate domain to two. For $N = 50$, we would consider the first 3000 eigenvalues. This is still a very densely populated grid that is burdened by the same challenges of the full spectrum. By incrementally reducing the number of eigenvalues considered, we find that the computed eigenvalues emanate from the origin and generate larger modulus eigenvalues (and presumably less accurate) eigenvalues with increases in N. The first eigenvalues and, presumably, the most accurately calculated, lie within a narrow band along the critical line. Furthermore, most of those are damped.



a) Chedevergne's results with respect to chamber length

b) Complex-lamellar results with respect to aspect ratio

Figure 6.37: Figure by Chedevergne [1] illustrating the effect of chamber length compared with current results for the complex-lamellar BV.

6.6.3 On Chamber Length

The streamfunction formulation by Chedevergne and coworkers requires them to define an extrapolating boundary condition at the exit plane [1, 123]. As previously discussed the present formulation makes use of a different physical requirement. Here, we apply an acoustically closed condition. This difference modifies the shape of the wave at the endwall. While the waves shown here (or their derivatives) are always forced to zero, those of Chedevergne and coworkers are not. They conclude that the wave is consistent for any chamber length and simply clipped at different points for different chamber lengths. This could be a direct result of an open aft end boundary condition rather than an acoustically closed condition or more importantly, this could easily be an artifact of an extrapolation at the endwall. Elsewhere in their discussion, they show that for varying chamber lengths, the amplification (ω_i) increases while ω_r associated with each eigenmode remains nearly unchanged for all chamber lengths. Chedevergne's result obtained for the Taylor-Culick

mean flow is depicted in Fig. 6.37a [123]. Similarly, selected spectra for the complex-lamellar bidirectional vortex are presented in Fig. 6.37b. Despite differences in formulation, we can also conclude that increasing the aspect ratio increases the amplification but leaves the frequency nearly unchanged. The large amount of overlap and scatter near and around the origin makes it difficult to confirm this for all frequencies, however, the region depicted here appears to share his conclusion. Similar correlations can be seen in the asymmetric spectra as well.

6.6.4 Quantifying Numerical Error

Our results can be numerically differentiated and backsubstituted into the governing equations as a means of error checking. Table 6.2 tabulates the maximum error incurred by backsubstituting for the contour plots shown in their respective sections. Recall that the actual error is likely better than those posted here since backsubstitution compounds the error of numerically differentiating the solution on top of the numeric error already incurred. Regardless, per Table 6.2, errors for the contour plots previously shown seem to be negligible and well within acceptable values to confirm the correctness of the given solutions.

Table 6.2: Backsubstitution and boundary condition error values for plotted asymmetric eigenvector examples.

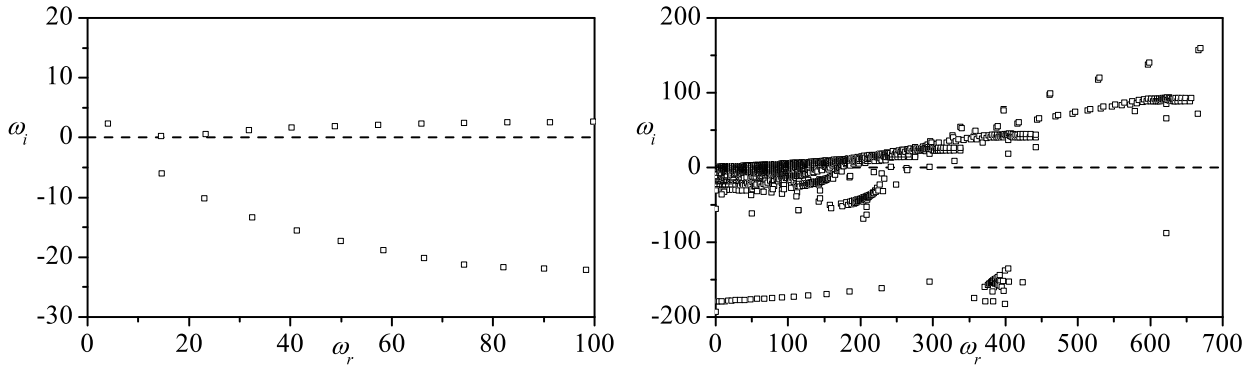
Example	Linearized N-S Equations				Boundary Conditions			
	Cont.	r-mom.	θ-mom.	z-mom.	$m(0,z)$	$m(1,z)$	$m(r,0)$	$m(r,l)$
CL	1.77E-012	6.20E-011	8.48E-011	9.49E-011				
$u_r(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$u_\theta(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$u_z(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$p(r,z)$					0.00E+00	3.18e-012*	3.58E-012*	1.19E-012*
LB	7.14E-013	2.1361E-011	1.46E-011	3.21E-011				
$u_r(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$u_\theta(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$u_z(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$p(r,z)$					0.00E+00	4.44E-013*	1.51E-012*	3.62E-013*
HB	1.45E-012	1.38E-011	1.34E-011	2.36E-011				
$u_r(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$u_\theta(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$u_z(r,z)$					0.00E+00	0.00E+00	0.00E+00	0.00E+00
$p(r,z)$					0.00E+00	9.21E-013*	1.35E-012*	1.06E-012*

*Derivative conditions are used along these boundaries.

Chapter 7

Conclusions

Both the LNP and biglobal analyses suggest an overall stable mean flow. This conclusion can only be made by considering the average growth rate near each discrete circular frequency. We observe that few unstable discrete frequencies exist. By decreasing the inlet parameter, κ, and inversely increasing the swirl number, the small number of unstable frequencies can be reduced or eliminated completely. While it is true that the hydrodynamic wave will grow if an unstable eigenvalue is excited, damped eigenvalues appear at similar circular frequencies. Very small deviations from an unstable eigenvalue will return the system to a stable state. Also, the overall growth rate for all circular frequencies is negative for both formulations. While this alone does not ensure stability, it may be an indicator. In contrast, the Taylor-Culick spectrum shown in Fig. 7.1 suggests a positive growth rate over a wide range of circular frequencies.



a) LNP spectrum for the Taylor-Culick model **b)** Biglobal spectrum for the Taylor-Culick model

Figure 7.1: Asymmetric spectral plots of the Taylor-Culick mean flow, with $q = 1$, $\alpha = 3$, $z = 1.5$, $Re = 10,000$.

7.1 Comparing the LNP and Biglobal Solutions

Since the size of the computed spectrum is drastically increased for the biglobal formulation, it is difficult to directly compare the spectral results of the LNP and biglobal methods. Furthermore, the LNP approach requires the specification of both an axial position and a spatial wave number whereas the biglobal method does not. Fortunately, neither axial position nor wave number greatly affects the spectrum. Figure 7.2 attempts to make a comparison near the origin for a constant Reynolds number. Two lines are overlayed to identify and compare dense spectral structures for each approach. Although, the biglobal approach does not render nearly as well defined spectral formations, a similar pseudo-continuous spectral line can be identified for the two figures. Both emanate from the origin and retain a downward inflection. Neither spectral line possess unstable eigenvalues but the biglobal line is flatter and, in general, less damped than the LNP line. Better agreement may be found by increasing the grid resolution for the biglobal formulation.

A more revealing comparison can be made between the waveforms. The biglobal solution identifies well defined oscillations near the headwall that are drawn from the outer region into the core and down the chamber. This behavior is somewhat apparent in the LNP solutions

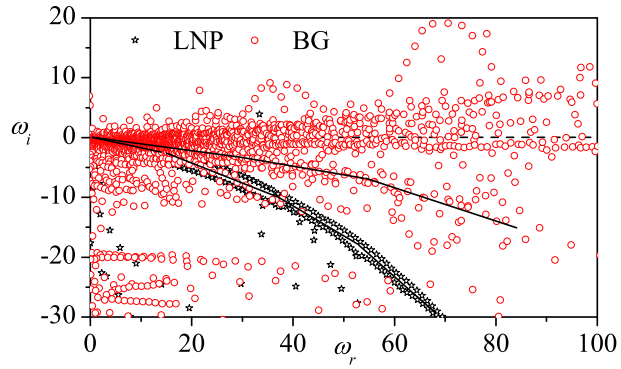
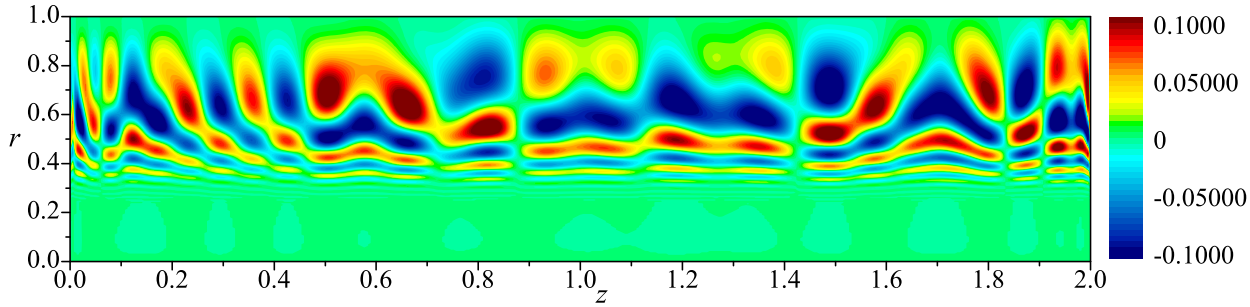


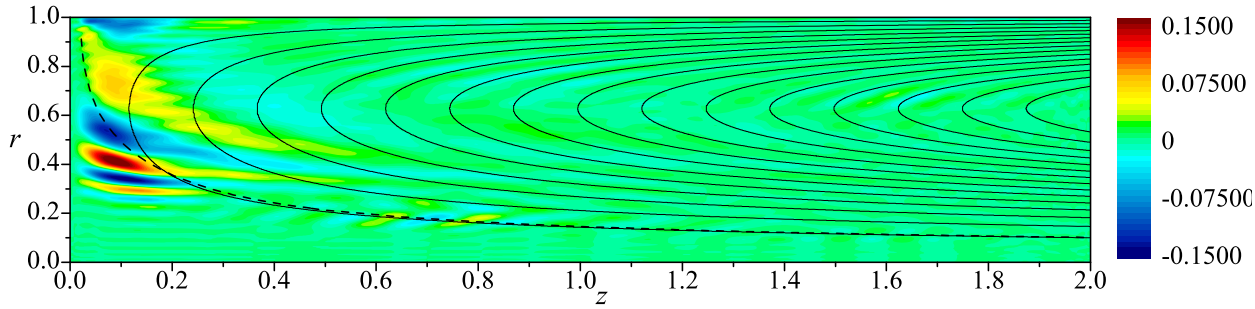
Figure 7.2: A comparison between the LNP and biglobal approach applied to the complex-lamellar mean flow with $\alpha = 3$, $z = 1.5$, $Re = 10,000$, and $\kappa = 0.1$.

but is much less defined. We see these oscillations most clearly in the linear Beltramian axial velocity profile. Figure 7.3 compares both LNP and biglobal axial contours for the first undamped eigenvalue. Off hand we see, the initial oscillation amplitudes are comparable for both solutions. This graph identifies three glaring differences. First, the LNP solution predicts the hydrodynamic fluctuation will persist throughout the length of the chamber while the biglobal solution shows no significant fluctuations past a distance of about $0.25l$. This characteristic of the biglobal contour plots suggests that the bidirectional vortex is spatially unstable as the outer vortex travels toward the headwall and spatially stable as the inner vortex travels downstream. Second, the fluctuations in the LNP solution penetrate radially from the sidewall a consistent radial distance throughout the length of the chamber. The headwall oscillations of the biglobal solution penetrate farther toward the centerline and are not a consistent distance from the sidewall. They are mated with oscillations that appear to emanate from the core vortex near the centerline to form a "V" at their interface. Lastly, the biglobal analysis suggests that oscillations occur around the streamlines of the mean flow. This pattern is not identifiable in the LNP analysis.

Being covered in passing only, the results obtained from the LNP formulation should be taken with caution for this class of mean flow solutions. Recall that this formulation is



a) LNP asymmetric axial velocity wave for the linear Beltramian model

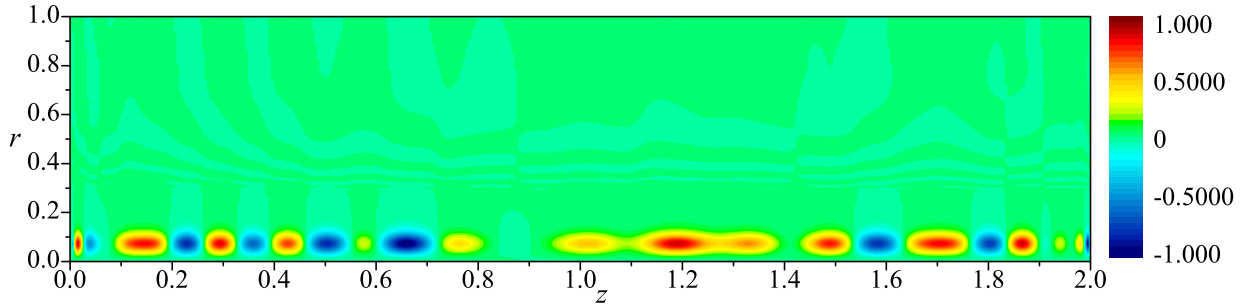


b) Biglobal asymmetric axial velocity wave for the linear Beltramian model

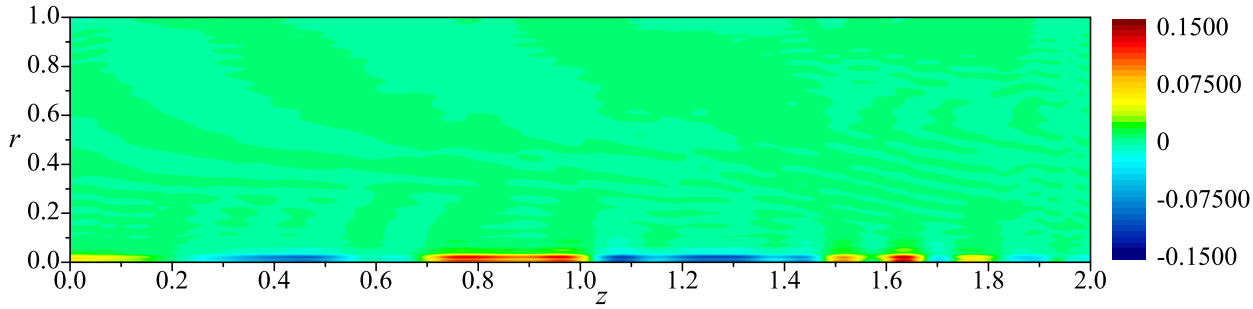
Figure 7.3: Comparison between the LNP and biglobal asymmetric axial velocity contour plots, with $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$.

in violation of the parallel-flow assumption and cannot properly analyze short, bidirectional vortex flowfields. In light of the strong physical arguments accompanying the biglobal results, solutions of this type are better predictors of physical hydrodynamic breakdown.

Similarities between the predicted waveforms in the radial and tangential directions are less evident. However, one notable similarity is found in the pressure fluctuation. While the majority of the chamber is steady, pressure fluctuations are confined to the centerline with oscillations occurring along the axial direction. The reader is referred to Fig. 7.4 for visual confirmation of this observation. As shown in the figure, oscillations predicted by the biglobal approach are closer to the centerline than those predicted by the LNP formulation. Also, the LNP solution suggests slightly higher amplitude fluctuations. This is likely a result of streamwise derivatives being omitted in favor of a discrete wave number. In either case, the pressure fluctuations appear in the region where the magnitude of mean flow pressure



a) LNP asymmetric pressure wave for the linear Beltramian model



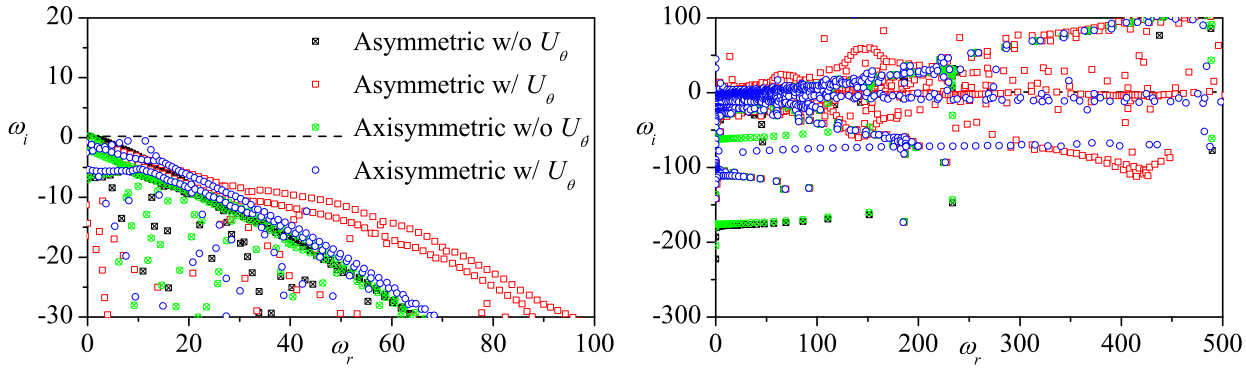
b) Biglobal asymmetric pressure wave for the linear Beltramian model

Figure 7.4: Comparison between the LNP and biglobal asymmetric pressure contour plots, with $q = 1$, $\alpha = 3$, $Re = 10,000$, and $\kappa = 0.1$.

is greatest. The perturbation is an insignificant component in the resulting instantaneous pressure. An attempt to determine the wave number in the LNP formulation that best agrees with the streamwise period in the biglobal solution would be an opportunity for further study.

7.1.1 The Effect of the Tangential Velocity

The inclusion of a tangential velocity is a unique aspect of this work. A high speed swirling velocity will introduce two obvious physical attributes to the problem: a shear layer near the centerline and centrifugal forces. While the new centerline shear layer will introduce vorticity and potentially instigate flow breakdown emanating from $r = 0$, the centrifugal forces will act to negate vortex generation along the sidewall. The ensuing behavior may warrant further investigation and parametrization to resolve its unique characteristics. In brief, Fig. 7.5 suggests that the pseudo-continuous spectral lines of both axisymmetric and



a) LNP spectral comparison **b)** BG spectral comparison

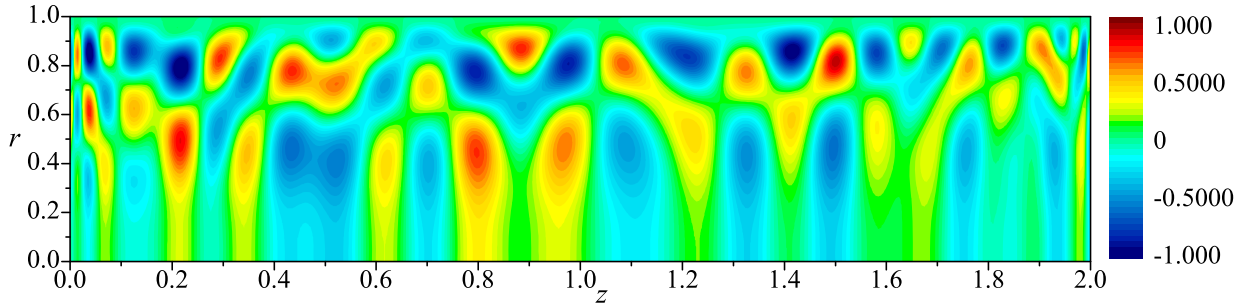
Figure 7.5: A comparison between the spectra of the linear Beltramian model to identify the effect of a swirl component in the mean flow. Here, $\alpha = 3$, $z = 2$, $Re = 10,000$, and $\kappa = 0.1$.

asymmetric models are nearly identical when swirl is not included. Both of these spectra resemble the axisymmetric solution with swirl quite closely. Recalling that $q = 0$ is commonly assumed from the onset, this close overlap suggests that this assumption is acceptable for two-dimensional base flows; however, the inclusion of a tangential velocity component changes the spectral characteristics for asymmetric flow. This is an expected behavior when reconsidering the stability equations. Nearly the same terms are negated when either $U_\theta = 0$ or $q = 0$; hence, the similarity in the spectral results. For similar reasons, the biglobal results show no significant difference between the cases of $q = 0$ and $q = 1$ when the tangential mean flow velocity is not included while the two spectra overlap well when U_θ is included. The greatest difference is the appearance of eigenvalues below $\omega_i = 0$ when U_θ is included. These eigenvalues are present for both the $q = 0$ and $q = 1$ cases. This is important when drawing conclusions about the overall stability. When the tangential velocity is included, the unstable eigenvalues at approximately $\omega_i \geq 300$ are paired with stable, damped eigenvalues in the same range. Together, the net result is stable. This is not the case when the tangential velocity is included.

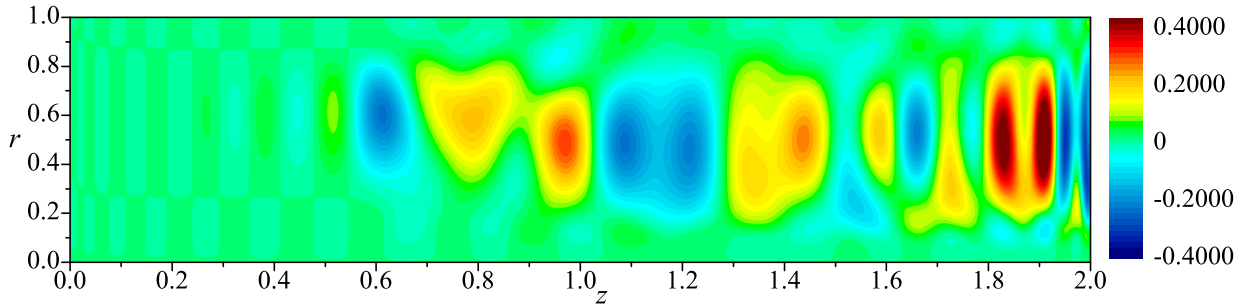
For the LNP waveforms shown in Fig. 7.6, we see a significant difference in the spatial shape of the axial solution compared to those in Fig. 5.18 and Fig. 5.20. For the axisymmetric

solution, we find much larger amplitude oscillations for the case without swirl included than previously reported. The overall waveform appears to be much more turbulent throughout the chamber length as well. However, the asymmetric solution is significantly different. The absence of a tangential base flow component simultaneously increases the oscillation amplitude while decreasing the degree of radial oscillations. The parietal form previously found gives way to a more volumetric breakdown across the radius. Even though the spectrum is not significantly different, the large initial amplitude suggests that the presence of a swirl velocity in the mean flow inhibits the transition to turbulence. Surprisingly, the axisymmetric solution is more turbulent. This is not seen elsewhere in this work, but is in agreement with similar solutions that employ the spatial theory of one and two-dimensional base flows. Similar results can be shown for u_r, u_q, and p. As before, the axisymmetric biglobal solution remains nearly zero throughout the domain. The asymmetric solution shows a distinct axial standing wave where the period is longest near the centerline and shortest at both the headwall and endwall. This is also seen in the LNP solution. The maximum amplitude is comparable to the solution with swirl included. The highest amplitude oscillations occur near the endwall rather than the headwall and there appears to be less distinct oscillation around the streamlines of the base flow. Close observation suggests the mantle separates two distinct instability regions. Radially, the wave changes sign near the mantle. This is consistent along the the length of chamber. While the spectra are similar, this analysis approach reaffirms previous arguments stating the overall stability is likely to improve when swirl exists. The waveforms suggest a more distinct axial wave when swirl is omitted but the amplitudes and growth rates are comparable.

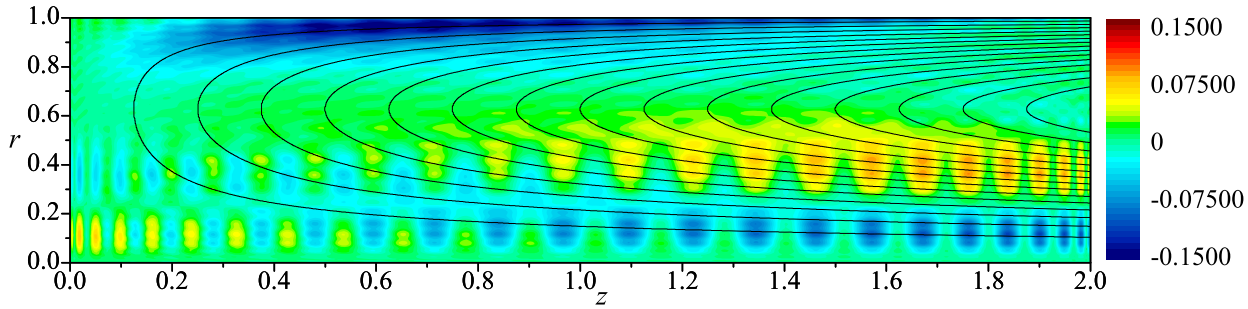
The results in the previous chapter where two values of the inflow parameter, κ, identify a more conclusive avenue to determine the contribution of tangential velocity on the stability. This parameter is related to the swirl intensity through the modified swirl number, σ. Increasing values of κ reflect a less swirl dominated flow. Decreasing κ intensifies the swirl and increases the centrifugal forces that tend to stabilize the flow in the chamber. It is



a) LNP axisymmetric axial wave for the linear Beltramian model with U_θ omitted



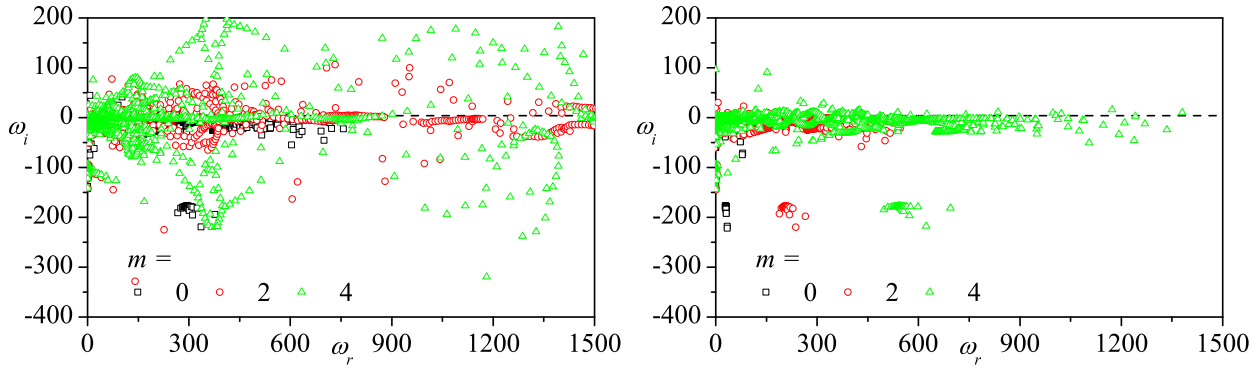
b) LNP asymmetric axial wave for the linear Beltramian model with U_θ omitted



c) BG asymmetric axial wave for the linear Beltramian model with U_θ omitted

Figure 7.6: Comparison between the linear Beltramian axial wave contour plots to illustrate the waveform when U_θ is omitted. Here, $\alpha = 0.5$ $Re = 10,000$, and $\kappa = 0.1$. The axisymmetric biglobal waveform is zero.

evident in the last chapter that decreasing κ acts to stabilize the flow. To further show this effect, we consider the effect of κ on multidirectional flow. This particular parameter shows the greatest scatter of eigenvalues. Specifically, the solutions reported for multidirectional flow of the linear Beltramian model in Fig. 6.21 suggested an increase in unstable eigenvalues with increasing flow reversals. Figure 7.7 identifies the spectrum for yet a smaller value of κ than reported in Ch. 6. In Fig. 7.7, we identify that while decreasing κ to 0.01 retains



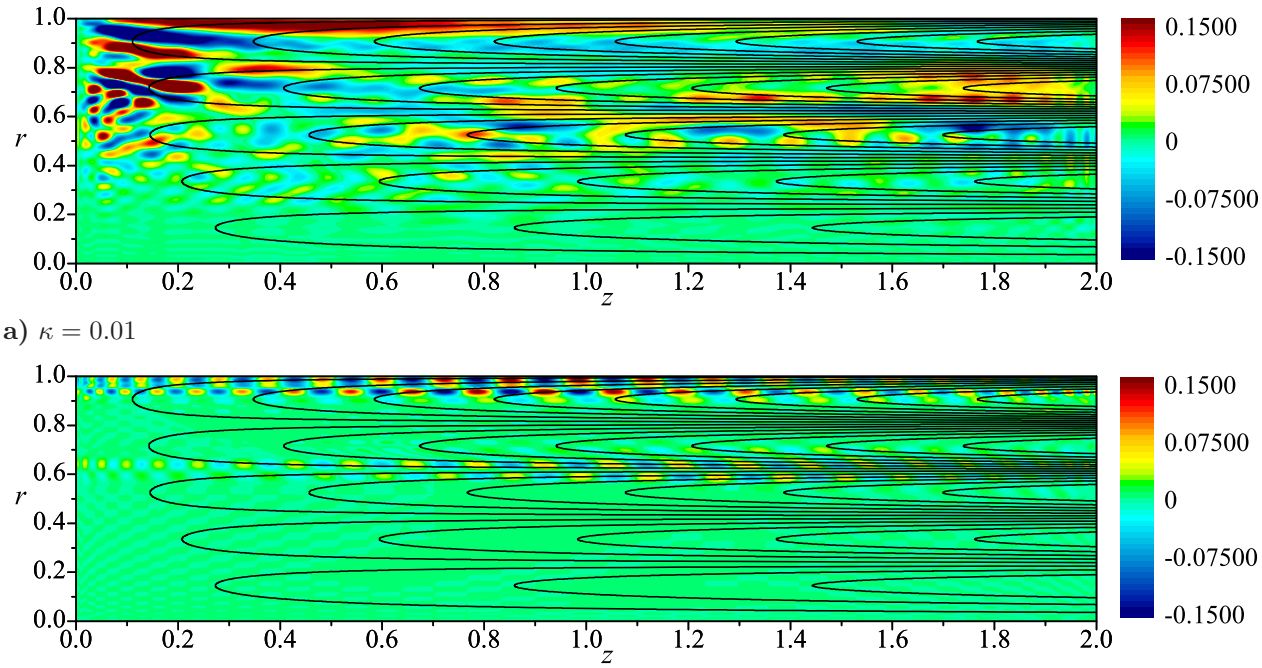
a) $\kappa = 0.01$ **b)** $\kappa = 0.001$

Figure 7.7: The multidirectional, linear Beltramian model for two values of κ. Here $q = 1$, $Re = 10,000$, and $l = 2$.

some amplified circular frequencies, $\kappa = 0.001$ drastically reduces the number of unstable eigenvalues. This is consistent with the hypothesis that increasing swirl intensity will increase stability. Likewise, axial contour plots for the first unstable eigenvalue are shown in Fig. 7.8. The maximum amplitude for $\kappa = 0.01$ is comparable to those shown for $\kappa = 0.1$. The amplitudes for $\kappa = 0.001$ are slightly lower. We also see that decreasing κ seems to force the oscillations closer to the wall. In the $\kappa = 0.001$ case, oscillations are only apparent in the region of negative axial mean flow with decreasing magnitude toward the centerline. Once again, we can conclude that increasing swirl does indeed have a stabilizing effect on the hydrodynamic predictions. This result is significant and suggests that inducing swirl will improve the hydrodynamic stability in propulsive devices.

7.2 Unstable Frequencies

The difficulty of identifying unstable circular frequencies amidst such densely populated spectra has been articulated in Ch. 6 and attempts at data reduction have been tabulated in App. C. For the sake of brevity only the first ten computed circular frequencies and their distributed growth rate are compiled for each plot. No unstable frequencies are predicted for



a) $\kappa = 0.01$

b) $\kappa = 0.001$

Figure 7.8: Comparison between the multidirectional linear Beltramian axial wave contour plots for two values of κ. Here $q = 1$, $Re = 10,000$, and $l = 2$.

any parametric result for $\omega_r < 60$ where the onset is predicted between 75 and 105 for most parameters studied. This is an indicator that the hydrodynamic breakdown is most prevalent in the high frequency domain. Several other interesting results can be identified within the tables. As we have already seen, a decrease in κ drastically improves the stability. Likewise, lowering the Reynolds number has a similar effect. Decreasing the chamber aspect ratio increases the overall stability but it is decreased by increasing the number of flow reversals. In this case, as m increases, instability is predicted for successively lower circular frequencies.

A particularly interesting result is shown in Table C.9 and Table C.14. For the case of $\kappa = 0.01$, many unstable eigenvalues are expected for $l = 2.5$ and 0.5 but only one is predicted for $l = 1.5$. It has previously been suggested that future studies should consider the stability near an aspect ratio of unity. Spectra for $l < 1$ appear to possess unique characteristics not seen for $l > 1$. It is apparent that the stability of short chambers is dictated primarily by

the tangential velocity. In long chambers, where the axial velocity is significantly larger at the exit plane, the axial velocity adds a greater contribution to the stability.

7.3 Eigensolver Implementation

In retrospect, accelerated convergence could be obtained with an Arnoldi type solver if one limits oneself to the first few eigenvalues only. Assuming that the first $N/3$ eigenvalues are physically viable, an Arnoldi solver can be set up in MATLAB with the command `[V Lam]=eigs(Amat,Bmat,round(N/3),0)`. This will calculate only the smallest $N/3$ eigenvalues. Initial tests suggest that this procedure could significantly reduce the run time without sparing accuracy. Of course, the trade-off is an incomplete spectrum: this being the original motivation away from an Arnoldi solver. The process could be automated by restarting the solver at successive circular frequencies but the calculation of a complete spectrum is likely to be as time consuming as the QZ or LZ methods. Implementation of an Arnoldi eigensolver may become necessary as the matrices become larger and more sophisticated to include equations that quantify particle effects, thermal properties, and combustion reactions.

7.4 Future Work

The future of this work is open. Biglobal instability is still in its infancy but will become more prevalent as high-speed computing resources become more widely available. At the time of this writing, the majority of research uses a streamfunction formulation rather than the velocity formulation used here. A comparative study should be conducted to validate both methods and help to reduce the clutter of the spectra in the velocity formulation. Since the streamfunction formulation requires significantly less computational time, it could be the superior method when applicable. For axisymmetric flows, a hybrid model could be erected.

Since the axial and radial velocities of the mean flow are decoupled from u_θ, a streamfunction representation is adequate to calculate the base flow. This formulation could be extrapolated to the axisymmetric stability equations as well. The mean flow streamfunction and tangential velocity would appear as coefficients in the stability equations. This would result in a system of equations with two equations and two unknowns (the hydrodynamic streamfunction, ψ, and the tangential fluctuation, u_θ). This would cut the matrix size in half and allow for finer resolution for a very specific base flow; namely, one similar to the bidirectional vortex. To the author's knowledge, mixing the streamfunction with a decoupled tangential velocity has not been attempted, nor has this type of formulation been validated.

Hydrodynamic breakdown is vorticity driven and, as such, justifies the omission of dilatational terms from the LNS equations. Compressibility could be easily included by amending the existing spectral equations. This would disallow a simple streamfunction formulation, but would not add any increased size to the operator matrices. In doing so, we could identify how compressibility affects turbulent breakdown. Similarly, the effect of particle entrainment would be an interesting stability study. Particles are assumed to have a damping effect on combustion instability and would likely have the same effect here. The inclusion of particles requires an additional equation beyond momentum and continuity. Another equation would increase the LNP matrices to $5N \times 5N$ and the biglobal matrices to $5N^2 \times 5N^2$. At present, this increase in matrix size would exceed the computational capacity of available high-end desktop computers. LNP solutions would still be tractable, although more time consuming than before. Ideally, thermal properties and chemical processes would be included to more completely model fully reacting flows.

Conventional one and two-dimensional hydrodynamic stability studies only predict standing waves evolving in time. Standing wave solutions are appropriate for modeling transition to turbulence in a simple pipe flow as was shown by Reynolds' classic experiment [37], however, traveling waves are an important component of combustion chamber dynamics. The results presented here show where circulation zones are likely to occur but do not track

the subsequent vortices as they translate downstream. The ability to track the traveling perturbations is beyond the scope of this formulation and would be worthwhile to pursue in future work. Likewise, interactions between the hydrodynamic instability and the vortico-acoustic wave would be beneficial to explore. It could be speculated that the formation of the acoustic wave and subsequent vortical boundary layer could initialize and/or accelerate the onset of hydrodynamic breakdown. It could be beneficial to define the base flow as the sum of the mean flow and the vortico-acoustic wave for comparison purposes. Unfortunately, without experimental data, these ideas cannot be fully substantiated.

Bibliography

[1] F. Chedevergne, G. Casalis, and T. Féraille, "Biglobal Linear Stability Analysis of the Flow Induced by Wall Injection," *Physics of Fluids*, vol. 18, p. 014103, 2006.

[2] W. Rankine, *A Manual of Applied Mechanics*. Griffin, 1877.

[3] H. Lamb, *Hydrodynamics*. Dover Publications, 1932.

[4] C. W. Oseen, "Über Wirbelbewegung in Einer Reibenden Flüssigkeit," *Arkiv foer Matematik, Astronomi, och Fysik*, vol. 7, pp. 1–13, 1911.

[5] J. Burgers, "A Mathematical Model Illustrating the Theory of Turbulence," *Advances in Applied Mechanics*, vol. 1, pp. 171–199, 1948.

[6] N. Rott, "On the Viscous Core of a Line Vortex," *Zeitschrift für Angewandte Mathematik und Physik (ZAMP)*, vol. 9, no. 5, pp. 543–553, 1958.

[7] G. Batchelor, "Axial flow in trailing line vortices," *Journal of Fluid Mechanics. n*, vol. 20, no. 04, pp. 645–658, 2006.

[8] J. W. Batterson, B. A. Maicke, and J. Majdalani, "Advancements in Theoretical Models of Confined Vortex Flowfields," in *2007 JANNAF Propulsion Conference*, JANNAF Paper TP-2007-222, (Denver, Colorado), May 2007.

[9] S. Alekseenko, P. Kuibin, V. Okulov, and S. Shtork, "Helical vortices in swirl flow," *Journal of Fluid Mechanics. n*, vol. 382, pp. 195–243, 1999.

[10] C. Eloy and S. Le Dizès, "Three-Dimensional Instability of Burgers and Lamb-Oseen Vortices in a Strain Field," *Journal of Fluid Mechanics*, vol. 378, pp. 145–166, 1999.

[11] P. Schmid and M. Rossi, "Three-dimensional stability of a Burgers vortex," *Journal of Fluid Mechanics. n*, vol. 500, pp. 103–112, 2004.

[12] C. Olendraru and A. Sellier, "Viscous effects in the absolute–convective instability of the Batchelor vortex," *Journal of Fluid Mechanics. n*, vol. 459, pp. 371–396, 2002.

[13] M. Perez-Saborid, M. Herrada, A. Gomez-Barea, and A. Barrero, "Downstream Evolution of Unconfined Vortices: Mechanical and Thermal Aspects," *Journal of Fluid Mechanics. n*, vol. 471, pp. 51–70, 2002.

[14] R. Sullivan, "A Two-Cell Vortex Solution of the Navier-Stokes Equations," *Journal of Aerospace Sciences*, vol. 26, pp. 767–768, 1959.

[15] J. Wu, "Conical Turbulent Swirling Vortex with Variable Eddy Viscosity," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 403, no. 1825, pp. 235–268, 1986.

[16] M. Bloor and D. Ingham, "The Flow in Industrial Cyclones," *Journal of Fluid Mechanics*, vol. 178, pp. 507–519, 1987.

[17] T. A. Barber and J. Majdalani, "Exact Eulerian Solution of the Conical Bidirectional Vortex," in *45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2009-5306, (Denver, Colorado), Aug. 2009.

[18] P. W. Gloyer, W. H. Knuth, and J. Goodman, "Overview of Initial Research into the Effects of Strong Vortex Flow on Hybrid Rocket Combustion and Performance," in *CSTAR Fifth Annual Symposium*, Paper N96-16953, (Tullahoma, Tennessee), Jan. 1993.

[19] W. Knuth, M. Chiaverini, J. Sauer, and D. Gramer, "Solid-Fuel Regression Rate Behavior of Vortex Hybrid Rocket Engines," *Journal of Propulsion and Power*, vol. 18, no. 3, pp. 600–609, 2002.

[20] M. Chiaverini, M. Malecki, J. Sauer, W. Knuth, and J. Majdalani, "Vortex Thrust Chamber Testing and Analysis for O2-H2 Propulsion Applications," in *39th*

AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, AIAA Paper 2003-4473, (Huntsville, Alabama), July 2003.

[21] A. Vyas and J. Majdalani, "Exact Solution of the Bidirectional Vortex," *AIAA Journal*, vol. 44, no. 10, p. 2208, 2006.

[22] J. Majdalani and S. Rienstra, "On the Bidirectional Vortex and Other Similarity Solutions in Spherical Coordinates," *Journal of Applied Mathematics and Physics (ZAMP)*, vol. 58, no. 2, pp. 289–308, 2007.

[23] J. Majdalani and M. J. Chiaverini, "On Steady Rotational Cyclonic Flows: The Viscous Bidirectional Vortex," *Physics of Fluids*, vol. 21, no. 10, p. 103603, 2009.

[24] J. W. Batterson and J. Majdalani, "Sidewall Boundary Layers of the Bidirectional Vortex," *Journal of Propulsion and Power*, vol. 26, no. 1, pp. 102–112, 2009.

[25] A. Vyas, J. Majdalani, and M. Chiaverini, "The Bidirectional Vortex. Part 1: An Exact Inviscid Solution," in *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2003-5052, p. 39, 2003.

[26] J. Majdalani, "Exact Eulerian Solutions of the Cylindrical Bidirectional Vortex," in *45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2009-5307, (Denver, Colorado), Aug. 2009.

[27] J. Wu, H. Ma, and M. Zhou, *Vorticity and Vortex Dynamics*. New York: Springer, New York, 2006.

[28] J. W. Batterson and J. Majdalani, "On the Viscous Bidirectional Vortex. Part 1: Linear Beltramian Motion," in *46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2010-6763, (Nashville, Tennessee), July 2010.

[29] J. W. Batterson and J. Majdalani, "On the Viscous Bidirectional Vortex. Part 2: Noninear Beltramian Motion," in *46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2010-6764, (Nashville, Tennessee), July 2010.

[30] A. Vyas, J. Majdalani, and M. Chiaverini, "The Bidirectional Vortex. Part 3: Multiple Solutions," in *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2003-5054, (Huntsville, Alabama), July 2003.

[31] M. Anderson, R. Valenzuela, C. Rom, R. Bonazza, and M. Chiaverini, "Vortex Chamber Flow Field Characterization for Gelled Propellant Combustor Applications," in *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2003-4474, (Huntsville, Alabama), p. 39, July 2003.

[32] C. J. Rom, M. H. Anderson, and M. J. Chiaverini, "Cold Flow Analysis of a Vortex Chamber Engine for Gelled Propellant Combustor Applications," in *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2004-3359, 2004.

[33] J. W. Batterson and J. Majdalani, "On the Viscous Bidirectional Vortex. Part 3: Multiple Mantles," in *46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2010-6765, (Nashville, Tennessee), July 2010.

[34] H. Helmholtz, "Über diskontinuierliche Flüssigkeits-Bewegungen," *Monatsberichte der Königlich Preussische Akademie der Wissenschaften zu Berlin*, pp. 215–228, 1868.

[35] W. Kelvin, "The Influence of Wind on Waves in Water Supposed Frictionless," *Philosophical Magazine*, vol. 4, no. 42, pp. 368–374, 1871.

[36] J. W. S. Rayleigh, "On the Stability, or Instability of Certain Fluid Motions," *Proceedings of the London Mathematical Society*, vol. XI, pp. 57–70, 1880.

[37] O. Reynolds, "An Experimental Investigation of the Circumstances which Determine whether the Motion of Water shall be Direct of Sinuous, and of the Law of Resistance in Parallel Channels," *Philosophical Transactions of the Royal Society of London*, vol. II, pp. 51–105, 1883.

[38] F. Chedevergne, G. Casalis, and J. Majdalani, "DNS investigation of the Taylor-Culick flow stability," in *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2007-5796, (Cincinnati, OH), 2007.

[39] P. Drazin, *Introduction to Hydrodynamic Stability*. Cambridge U. Press, 2002.

[40] V. Theofilis and S. J. Sherwin, "Global Instabilities in Trailing Edge Laminar Separated Flow on a N.A.C.A. 0012 Aerofoil," in *15th International Symposium on Airbreathing Engines*, (Bangalore, India), 2001.

[41] V. Theofilis, "Advances in Global Linear Instability Analysis of Nonparallel and Three-Dimensional Flows," *Progress in Aerospace Sciences*, vol. 39, no. 4, pp. 249–316, 2003.

[42] D. Barkley, G. Gomes, and R. Henderson, "Three-Dimensional Instability in a Flow Over a Backward Facing Step," *Journal of Fluid Mechanics*, vol. 473, pp. 167–190, 2002.

[43] H. Stüer, *Investigation of Separation on a Forward Facing Step*. PhD thesis, Swiss Federal Institute of Technology Zürich, 1999.

[44] H. Stüer, A. Gyr, and W. Kinzelbach, "Laminar-Turbulent Transition of a Separating Flow on a Forward Facing Step," in *Proceedings of the IUTAM Laminar-Turbulent Symposium*, (Sedona, AZ, USA), pp. 541–546, 2000.

[45] B. L. Jensen, B. M. Sumer, and J. Fredsoe, "Turbulent Oscillatory Boundary Layers at High Reynolds Numbers," *Journal of Fluid Mechanics*, vol. 206, no. -1, pp. 265–297, 1989.

[46] P. Merkli and H. Thomann, "Transition to Turbulence in Oscillating Pipe Flow," *Journal of Fluid Mechanics*, vol. 68, no. 03, pp. 567–576, 1975.

[47] S. I. Sergeev, "Fluid Oscillations in Pipes at Moderate Reynolds Numbers," *Fluid Dynamics*, vol. 1, pp. 121–122, 1966.

[48] S. Uchida, "The Pulsating Viscous Flow Superposed on the Steady Laminar Motion of Incompressible Fluid in a Circular Pipe," *Zeitschrift fr Angewandte Mathematik und Physik (ZAMP)*, vol. 7, pp. 403–422, 1956. 10.1007/BF01606327.

[49] J. Majdalani, *Improved Flowfield Models in Rocket Motors and the Stokes Layer with Sidewall Injection*. PhD thesis, The University of Utah, 1995.

[50] G. A. Flandro, "On Flow Turning," in *AIAA 31st Joint Propulsion Conference*, 95-2730, 1995.

[51] J. Majdalani and G. A. Flandro, "The oscillatory pipe flow with arbitrary wall injection," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 458, no. 2023, pp. 1621–1651, 2002.

[52] T. Jankowski and J. Majdalani, "Vortical and acoustical mode coupling inside a porous tube with uniform wall suction," *The Journal of the Acoustical Society of America*, vol. 117, pp. 3448–3458, 2005.

[53] J. Majdalani and S. Rienstra, "Two asymptotic forms of the rotational solution for wave propagation inside viscous channels with transpiring walls," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 55, no. 1, pp. 141–162, 2002.

[54] J. Majdalani, "Multiple Asymptotic Solutions for Axially Travelling Waves in Porous Channels," *Journal of Fluid Mechanics*, vol. 636, pp. 59–89, 2009.

[55] G. Casalis and F. Vuillot, "Motor Flow Instabilities - Part 2 Intrinsic Linear Stability of the Flow Induced by Wall Injection," tech. rep., ONERA, 2002.

[56] M. Gaster, "A Note on the Relation Between Temporally-Increasing and Spatially-Increasing Disturbances in Hydrodynamic Stability," *Journal of Fluid Mechanics. n*, vol. 14, no. 02, pp. 222–224, 1962.

[57] J. Griffond and G. Casalis, "On the Dependence on the Formulation of Some Nonparallel Stability Approaches Applied to the Taylor Flow," *Physics of Fluids*, vol. 12, p. 466, 2000.

[58] P. Andersson, "Modelling of Boundary Layer Stability," tech. rep., Royal Institute of Technology, 1999.

[59] C. Airiau and G. Casalis, "Stabilité Linéaire de la Couche Limite par un Système d'Éuestions Parabolique," *La Rechereche Aérospatiale*, vol. 5, 1993.

[60] M. Langlois, G. Casalis, and D. Arnal, "On the Practical Application of the P.S.E. Approach to Linear Stability Analysis," *Journal of Aerospace Science and Technology*, vol. 2, no. 3, pp. 167–176, 1998.

[61] H. Haj-Hariri, "Characteristics Analysis of the Parabolized Stability Equations," *Studies in Applied Mathematics*, vol. 92, pp. 41–53, 1994.

[62] F. Li and M. R. Malik, "Mathematical Nature of Parabolized Stability Equations," in *Laminar-Turbulent Transition. Proc. 4th IUTAM Symposium*, pp. 205–212, 1995.

[63] F. Li and M. R. Malik, "On the Nature of the PSE Approximation," *Theoretical and Computational Fluid Dynamics*, vol. 8, pp. 253–273, 1996.

[64] P. Andersson, D. S. Henningson, and A. Hanifi, "On the Stabilization Procedure for the Parabolic Stability Equations," *Journal of Engineering Mathematics*, vol. 33, pp. 311–332, 1998.

[65] T. Herbert, "Parabolized Stability Equations," *Annual Review of Fluid Mechanics*, vol. 29, no. 1, pp. 245–283, 1997.

[66] C. Yen and N. Messersmith, "Application of Parabolized Stability Equations to the Prediction of Jet Instabilities," *AIAA Journal*, vol. 36, no. 8, pp. 1541–1544, 1998.

[67] S. R. Lin and M. R. Malik, "On the Stability of Attachment-Line Boundary Layers. Part 1. The Incompressible Swept Hiemenz Flow," *Journal of Fluid Mechanics*, vol. 311, pp. 239–255, 1996.

[68] C. B. Moler and G. W. Stewart, "An Algorithm for Generalized Matrix Eigenvalue Problems," *SIAM Journal on Numerical Analysis*, vol. 10, pp. 241–256, 1973.

[69] C. B. Moler and G. W. Stewart, "An Algorithm for the Generalized Matrix Eigenvalue Problem $Ax = \lambda Bx$," tech. rep., Stanford, 1971.

[70] L. Kaufman, "Algorithm 496: The LZ Algorithm to Solve the Generalized Eigenvalue Problem for Complex Matrices," *ACM Transactions on Mathematical Software*, vol. 1, no. 3, pp. 271–281, 1975.

[71] L. Kaufman, "The LZ-Algorithm to Solve the Generalized Eigenvalue Problem," *SIAM Journal on Numerical Analysis*, vol. 11, pp. 997–1024, 1974.

[72] L. Kaufman, "A Generalization of the LR Algorithm to Solve AX=Lambda BX," tech. rep., Stanford, 1972.

[73] V. Theofilis, "On the Spatial Structure of Global Linear Instabilities and their Experimental Identification," *Journal of Aerospace Science and Technology*, vol. 4, no. 4, pp. 249–262, 2000.

[74] R. Heeg and B. Geurts, "Spatial Instabilities of the Incompressible Attachment-Line Flow Using Sparse Matrix Jacobi-Davidson Techniques," *Applied Scientific Research*, vol. 59, pp. 315–329, 1998.

[75] J. Griffond, "Receptivity and aeroacoustic resonance in channels with blowing walls," *Physics of Fluids*, vol. 14, p. 3946, 2002.

[76] E. Abu-Irshaid, J. Majdalani, and G. Casalis, "Hydrodynamic Stability of Rockets with Headwall Injection," *Physics of Fluids*, vol. 19, p. 024101, 2007.

[77] J. Majdalani and T. Saad, "The Taylor-Culick Profile with Arbitrary Headwall Injection," *Physics of Fluids*, vol. 19, p. 093601, 2007.

[78] M. Bertato, D. Giaiotti, A. Manzato, and F. Stel, "An Interesting Case of Tornado in Friuli-Northeastern Italy," *Atmospheric Research*, vol. 67, pp. 3–21, 2003.

[79] K. Mallen, M. Montgomery, and B. Wang, "Reexamining the Near-Core Radial Structure of the Tropical Cyclone Primary Circulation: Implications for Vortex Resiliency," *Journal of the Atmospheric Sciences*, vol. 62, no. 2, pp. 408–425, 2005.

[80] W. Devenport, M. Rife, S. Liapis, and G. Follin, "The Structure and Development of a Wing-Tip Vortex," *Journal of Fluid Mechanics. n*, vol. 312, pp. 67–106, 2006.

[81] J. M. Burgers, "On the Resistance of Fluids and Vortex Motion," *Koninklijke Nederlandsche Akademie van Wetenschappen Proceedings*, vol. 23, no. 1, pp. 774–782, 1921.

[82] N. Rott, "High Speed Aerodynamics and Jet Propulsion - Theory of time-dependent laminar flows," in *Theory of Laminar Flows* (F. Moore, ed.), vol. IV, Princeton University Press, 1964.

[83] N. Rott, "Boundary Layers and their Interactions in Rotating Flows," *Progress in Aerospace Sciences*, vol. 7, pp. 111–144, 1966.

[84] J. W. Batterson, "Investigation of the Sidewall Boundary Layers in the Bidirectional Vortex Liquid Rocket Engine," Master's thesis, The University of Tennessee Space Institute, 2007.

[85] F. Culick, "Rotational Axisymmetric Mean Flow and Damping of Acoustic Waves in a Solid Propellant Rocket," *AIAA Journal*, vol. 4, no. 8, pp. 1462–1464, 1966.

[86] J. Majdalani, "Vortex Injection Hybrid Rockets," in *Fundamentals of Hybrid Rocket Combustion and Propulsion* (K. Kuo and M. J. Chiaverini, eds.), Progress in Astronautics and Aeronautics, ch. Chap. 6, pp. 247–276, Washington, DC: AIAA Progress in Astronautics and Aeronautics, 2007. NSF.

[87] L. Prandtl, "Zur Berechnung der Grenzschichten," *Zeitschrift für angewandte Mathematik und Mechanik (ZAMM)*, vol. 18, no. 1, pp. 77–82, 1938.

[88] H. Schlichting, "Lecture Series: Boundary Layer Theory Part 1 - Laminar Flow," tech. rep., NACA, 1949.

[89] N. Tetervin, "Boundary-Layer Momentum Equations for Three-dimensional Flow," tech. rep., National Advisory Committee for Aeronautics, 1947.

[90] A. H. Nayfeh, *Perturbation Methods*. Wiley, 1973.

[91] A. Erdélyi, *Asymptotic Expansions*. Dover publications, 1956.

[92] S. L. Bragg and W. R. Hawthorne, "Some Exact Solutions of the Flow Through Annular Cascade Actuator Discs," *Journal of the Aeronautical Sciences*, vol. 17, p. 243, 1950.

[93] A. Murray, A. Gudgen, M. Chiaverini, J. Sauer, and W. Knuth, "Numerical Code Development for Simulating Gel Propellant Combustion Processes," in *JANNAF*, 2004.

[94] R. L. Burden and J. D. Faires, *Numerical Analysis*. PWS-KENT Publishing Company, Boston, USA, 1989.

[95] L. N. Trefethen, *Spectral Methods in Matlab*. Society for Industrial Mathematics, 2000.

[96] G. Arfken, G. Arfken, and H. Weber, *Mathematical Methods for Physicists*. Academic Press, 2001.

[97] J. Weideman and S. Reddy, "A MATLAB Differentiation Matrix Suite," *ACM Transactions on Mathematical Software*, vol. 26, no. 4, p. 519, 2000.

[98] P. Henrici, *Essentials of Numerical Analysis With Pocket Calculator Demonstrations*. John Wiley & Sons, Inc., 1982.

[99] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*. Springer Verlag, 2007.

[100] B. Fornberg, "Generation of Finite Difference Formulas on Arbitrarily Spaced Grids," *Mathematics of computation*, vol. 51, no. 184, pp. 699–706, 1988.

[101] R. Voigt, D. Gottlieb, and M. Hussaini, *Spectral Methods for Partial Differential Equations*. Society for Industrial & Applied Mathematics, 1984.

[102] D. Gottlieb and L. Lustman, "The DuFort-Frankel Chebyshev Method for Parabolic Initial Boundary Value Problems," *Computers and Fluids*, vol. 11, no. 2, pp. 107–120, 1983.

[103] B. D. Welfert, "Generation of Pseudospectral Differentiation Matrices I," *SIAM Journal on Numerical Analysis*, vol. 34, no. 4, pp. 1640–1657, 1997.

[104] E. M. Elbarbary and S. M. El-Sayed, "Higher Order Pseudospectral Differentiation Matrices," *Applied Numerical Mathematics*, vol. 55, no. 4, pp. 425–438, 2005.

[105] T. Braconnier, V. Fraysse, and J. Rioual, "ARNCHEB users' guide: Solution of Large Non Symmetric or Non Hermitian Eigenvalue Problems by the Arnoldi-Tchebycheff Method," tech. rep., University of Minnesota, 1997.

[106] R. J. Radke, "A Matlab Implementation of the Implicitly Restarted Arnoldi Method for Solving Large-Scale Eigenvalue Problems," Master's thesis, Rice University, 1996.

[107] J. Scott, "An Arnoldi Code for Computing Selected Eigenvalues of Sparse, Real, Unsymmetric Matrices," *ACM Transactions on Mathematical Software*, vol. 21, no. 4, pp. 432–475, 1995.

[108] D. Sorensen, "Implicit Application of Polynomial Filters in a k-Step Arnoldi method," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 1, pp. 357–385, 1992.

[109] D. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2007.

[110] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Programming*. Cambridge U. Press, 1992.

[111] D. Lemonnier and P. Van Dooren, "Balancing Regular Matrix Pencils," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 1, pp. 253–263, 2007.

[112] J. Wilkinson and C. Reinsch, *Handbook for Automatic Computation.-Vol. 2: Linear Algebra*. Springer, 1971.

[113] R. C. Ward, "Balancing the Generalized Eigenvalue Problem," *SIAM Journal on Scientific and Statistical Computing*, vol. 2, no. 2, pp. 141–152, 1981.

[114] B. Parlett and C. Reinsch, "Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors," *Numerische Mathematik*, vol. 13, no. 4, pp. 293–304, 1969.

[115] J. C. F. Francis, "The QR Transformation-Part 2," *The Computer Journal*, vol. 4, pp. 332–345, 1962. article.

[116] J. C. F. Francis, "The QR Transformation: A Unitary Analogue to the LR Transformation-Part 1," *The Computer Journal*, vol. 4, pp. 265–272, 1961. article.

[117] H. Rutishauser, "Solution of Eigenvalue Problems with the LR Transformation," *National Bureau Standards Applied Mathematics Series*, vol. 49, pp. 47–81, 1958. article.

[118] J. Wilkinson, *The Algebraic Eigenvalue Problem*. London: Oxford University Press, 1965.

[119] E. Abu-Irshaid, J. Majdalani, and G. Casalis, "Hydrodynamic Instability of the Bidirectional Vortex," in *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2005-4531, (Tuscon, Arizona), July 2005.

[120] P. Drazin and W. Reid, *Hydrodynamic Stability*. Cambridge University, 1985.

[121] M. Gaster, "The Growth of Three-Dimensional Disturbances in Inviscid Flows," *Journal of Fluid Mechanics*, vol. 43, no. 04, pp. 837–839, 1970.

[122] J. C. French and J. Majdalani, "Hydrodynamic Stability Analysis of Solid Rocket Motors with Arbitrary Grain Design," in *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2007-5808, 2007.

[123] F. Chedevergne, *Instabilités Intrinsèques des Moteurs à Propergol Solide*. PhD thesis, L'école Nationale Supérieure de L'aéronautique et de L'espace, Sept 21, 2007 2007.

[124] C. Robitaillié-Montané and G. Casalis, "Méthode de Collocation Spectrale Appliquée à un Problème de Stabilité donné sous Forme d'équations aux Dérivées Partielles," Tech. Rep. RT-1/07895 DMAE, ONERA, March 2003.

[125] E. Merzari, S. Wang, H. Ninokata, and V. Theofilis, "Biglobal Linear Stability Analysis for the Flow in Eccentric Annular Channels and a Related Geometry," *Physics of Fluids*, vol. 20, p. 114104, 2008.

Appendix A

Derivations

A.1 Deriving the 1-D Cylindrical LNP Equations

The LNP equations are given by Eqs. (1.12a–1.12d) where they are presented for an incompressible, viscous, non-reacting flow. We have *Continuity:*

$$\frac{\partial \breve{u}_r}{\partial r} + \frac{\breve{u}_r}{r} + \frac{1}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\partial \breve{u}_z}{\partial z} = 0 \tag{A.1a}$$

Radial momentum:

$$\begin{aligned}\frac{\partial \breve{u}_r}{\partial t} + U_r\frac{\partial \breve{u}_r}{\partial r} + \breve{u}_r\frac{\partial U_r}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_r}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_r}{\partial \theta} - \frac{2U_\theta \breve{u}_\theta}{r} + U_z\frac{\partial \breve{u}_r}{\partial z} + \breve{u}_z\frac{\partial U_r}{\partial z} + \frac{\partial \breve{p}}{\partial r} \\ = \varepsilon\left(\frac{\partial^2 \breve{u}_r}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_r}{\partial r} - \frac{\breve{u}_r}{r^2} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_r}{\partial \theta^2} - \frac{2}{r^2}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\partial^2 \breve{u}_r}{\partial z^2}\right)\end{aligned} \tag{A.1b}$$

Tangential momentum:

$$\begin{aligned}\frac{\partial \breve{u}_\theta}{\partial t} + U_r\frac{\partial \breve{u}_\theta}{\partial r} + \breve{u}_r\frac{\partial U_\theta}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_\theta}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r \breve{u}_\theta}{r} + \frac{\breve{u}_r U_\theta}{r} + U_z\frac{\partial \breve{u}_\theta}{\partial z} + \breve{u}_z\frac{\partial U_\theta}{\partial z} + \frac{1}{r}\frac{\partial \breve{p}}{\partial \theta} \\ = \varepsilon\left(\frac{\partial^2 \breve{u}_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_\theta}{\partial r} - \frac{\breve{u}_\theta}{r^2} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_\theta}{\partial \theta^2} + \frac{2}{r^2}\frac{\partial \breve{u}_r}{\partial \theta} + \frac{\partial^2 \breve{u}_\theta}{\partial z^2}\right)\end{aligned} \tag{A.1c}$$

Axial momentum:

$$\begin{aligned}\frac{\partial \breve{u}_z}{\partial t} + U_r\frac{\partial \breve{u}_z}{\partial r} + \breve{u}_r\frac{\partial U_z}{\partial r} + \frac{U_\theta}{r}\frac{\partial \breve{u}_z}{\partial \theta} + \frac{\breve{u}_\theta}{r}\frac{\partial U_z}{\partial \theta} + U_z\frac{\partial \breve{u}_z}{\partial z} + \breve{u}_z\frac{\partial U_z}{\partial z} + \frac{\partial \breve{p}}{\partial z} \\ = \varepsilon\left(\frac{\partial^2 \breve{u}_z}{\partial r^2} + \frac{1}{r}\frac{\partial \breve{u}_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \breve{u}_z}{\partial \theta^2} + \frac{\partial^2 \breve{u}_z}{\partial z^2}\right)\end{aligned} \tag{A.1d}$$

An arbitrary fluctuation can be further defined to assume the form of the one-dimensional normal mode such that

$$\breve{m}(r,\theta,z,t) = m(r)\exp[i(q\theta + \alpha z - \omega t)] \tag{A.2}$$

In general $m(r)$ is a complex amplitude function representing the amplitude of oscillations. The tangential wave number, q, must take integer values. Although it is often set to zero at the onset of analysis, we will include it for completeness. The wave number, α, and the frequency, ω, are further identified as the combination of a real and imaginary component

$$\alpha = \alpha_r + i\alpha_i; \quad \omega = \omega_r + i\omega_i \tag{A.3}$$

where α_r is the longitudinal wave number, and ω_r is the circular frequency. Simple factoring of Eq. (A.2) clearly shows that the amplitude growth depends on ω_i with respect to time and α_i with respect to z:

$$\breve{m}(r,\theta,z,t) = m(r)e^{-\alpha_i z+\omega_i t}e^{i(q\theta+\alpha_r z-\omega_r t)} \tag{A.4}$$

Either a positive ω_i or a negative α_i would signal wave amplification in time or space, respectively. The spatial period is then dictated by α_r and the temporal frequency by ω_r.

Making these substitutions into Eqs. (1.9a–1.9d) results in the cylindrical *Local nonparallel* (LNP) equations for one-dimensional stability analysis. We find

Continuity:

$$\frac{\mathrm{d}u_r}{\mathrm{d}r} + \frac{u_r}{r} + iq\frac{u_\theta}{r} + i\alpha u_z = 0 \tag{A.5a}$$

Radial momentum:

$$\begin{aligned} -i\omega u_r + U_r\frac{\mathrm{d}u_r}{\mathrm{d}r} + u_r\frac{\partial U_r}{\partial r} + iq\frac{U_\theta u_r}{r} + \frac{u_\theta}{r}\frac{\partial U_r}{\partial \theta} - 2\frac{U_\theta u_\theta}{r} + i\alpha U_z u_r + u_z\frac{\partial U_r}{\partial z} + \frac{\mathrm{d}p}{\mathrm{d}r} \\ = \varepsilon\left(\frac{\mathrm{d}^2u_r}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}u_r}{\mathrm{d}r} - \frac{u_r}{r^2} - q^2\frac{u_r}{r^2} - 2iq\frac{u_\theta}{r^2} - \alpha^2 u_r\right) \end{aligned} \tag{A.5b}$$

Tangential momentum:

$$-i\omega u_\theta + U_r \frac{\mathrm{d}u_\theta}{\mathrm{d}r} + u_r \frac{\partial U_\theta}{\partial r} + iq\frac{U_\theta u_\theta}{r} + \frac{u_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r u_\theta}{r} + \frac{u_r U_\theta}{r} + i\alpha U_z u_\theta + u_z \frac{\partial U_\theta}{\partial z} + iq\frac{p}{r}$$
$$= \varepsilon \left(\frac{\mathrm{d}^2 u_\theta}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}u_\theta}{\mathrm{d}r} - \frac{u_\theta}{r^2} - q^2 \frac{u_\theta}{r^2} + 2iq\frac{u_r}{r^2} - \alpha^2 u_\theta \right) \quad \text{(A.5c)}$$

Axial momentum:

$$-i\omega u_z + U_r \frac{\mathrm{d}u_z}{\mathrm{d}r} + u_r \frac{\partial U_z}{\partial r} + iq\frac{U_\theta u_z}{r} + \frac{u_\theta}{r}\frac{\partial U_z}{\partial \theta} + i\alpha U_z u_z + u_z \frac{\partial U_z}{\partial z} + i\alpha p$$
$$= \varepsilon \left(\frac{\mathrm{d}^2 u_z}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}u_z}{\mathrm{d}r} - q^2 \frac{u_z}{r^2} - \alpha^2 u_z \right) \quad \text{(A.5d)}$$

These equations have no assumptions beyond the stability theory applied. They allow for nonsymmetric, fully three dimensional, base flow with incompressible instabilities in all three directions.

A.2 Deriving the Cylindrical Biglobal Stability Equations

In general, the derivation of the biglobal stability equations follows that of the LNP equations with the exception of the inclusion of a two-dimensional, biglobal, normal mode rather than the traditional one-dimensional form. In fact, we can begin with Eqs. (1.12a–1.12d) and consider a modal ansatz of the form

$$\breve{m} = m\left(r, z\right) \exp\left[i\left(q\theta - \omega t\right)\right] \quad \text{(A.6)}$$

To understand the applicability of this type of modal decomposition, we consider the parallel flow assumption as it pertains to classic, one-dimensional stability given by Eq. (1.14).

Accordingly, we assume a weak (or no) dependence of the base flow on the axial and tangential coordinates. This situation represents one extreme. The other being no assumptions about the base flow by allowing the amplitude function to be dependent on all three coordinates. For most analytic base flows, this is not necessary. For instance, the Taylor-plane or Taylor-Culick flows are two-dimensional and, as such, a two-dimensional perturbation becomes a natural assumption. Biglobal theory lies between the two extremes. The criterion for parallel-flow is modified to include streamwise variations. Here we assume

$$\frac{\partial M}{\partial \theta} \ll \frac{\partial M}{\partial r} \quad \text{and} \quad \frac{\partial M}{\partial \theta} \ll \frac{\partial M}{\partial z} \tag{A.7}$$

At the outset, the variation in the third direction becomes small compared to the first two [41]. Here we also insist that perturbations in the third direction are periodic given that q is an integer mode number. In the limit as $\partial M/\partial z \to 0$, the one-dimensional normal mode analysis is recovered. This confirms that the LNP stability equations are a subset of a larger system of partial differential equations.

Since the normal mode no longer includes a spatial wave number, we no longer have a "spatial" theory similar to the one-dimensional approach. Rather, we have a fully computed, two-dimensional spatial waveform. This approach captures spatial instability globally and, hence, is more illuminating as to the spatial character of the hydrodynamic wave. It is important to realize that a waveform with an initially high amplitude and low growth rate may be as destructive to the instantaneous velocity as a small amplitude wave with high growth rate. Historically, the physical waveform is neglected in favor of considering only the growth rate. The lack of a clear spatial theory forces us to reconsider the balance between waveform and growth rate when comparing one and two-dimensional results. We do, however, retain the familiar temporal theory by defining $\omega = \omega_r + i\omega_i$. Then the normal mode takes the form

$$\breve{m}(r, \theta, z, t) = m(r) e^{\omega_i t} e^{i(q\theta - \omega_r t)} \tag{A.8}$$

To make progress, we apply the biglobal ansatz to return our general biglobal stability equations. After much scrutiny we find

Continuity:

$$\frac{\partial u_r}{\partial r} + \frac{u_r}{r} + iq\frac{u_\theta}{r} + \frac{\partial u_z}{\partial z} = 0 \tag{A.9a}$$

Radial momentum:

$$\begin{aligned} -i\omega u_r + U_r\frac{\partial u_r}{\partial r} + u_r\frac{\partial U_r}{\partial r} + iq\frac{U_\theta u_r}{r} + \frac{u_\theta}{r}\frac{\partial U_r}{\partial \theta} - \frac{2U_\theta u_\theta}{r} + U_z\frac{\partial u_r}{\partial z} + u_z\frac{\partial U_r}{\partial z} + \frac{\partial p}{\partial r} \\ = \varepsilon\left(\frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{u_r}{r^2} - \frac{q^2}{r^2}u_r - \frac{2iq}{r^2}u_\theta + \frac{\partial^2 u_r}{\partial z^2}\right) \end{aligned} \tag{A.9b}$$

Tangential momentum:

$$\begin{aligned} -i\omega u_\theta + U_r\frac{\partial u_\theta}{\partial r} + u_r\frac{\partial U_\theta}{\partial r} + iq\frac{U_\theta u_\theta}{r} + \frac{u_\theta}{r}\frac{\partial U_\theta}{\partial \theta} + \frac{U_r u_\theta}{r} + \frac{u_r U_\theta}{r} + U_z\frac{\partial u_\theta}{\partial z} + u_z\frac{\partial U_\theta}{\partial z} + iq\frac{p}{r} \\ = \varepsilon\left(\frac{\partial^2 u_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial u_\theta}{\partial r} - \frac{u_\theta}{r^2} - \frac{q^2}{r^2}u_\theta + \frac{2iq}{r^2}u_r + \frac{\partial^2 u_\theta}{\partial z^2}\right) \end{aligned} \tag{A.9c}$$

Axial momentum:

$$\begin{aligned} -i\omega u_z + U_r\frac{\partial u_z}{\partial r} + u_r\frac{\partial U_z}{\partial r} + iq\frac{U_\theta u_z}{r} + \frac{u_\theta}{r}\frac{\partial U_z}{\partial \theta} + U_z\frac{\partial u_z}{\partial z} + u_z\frac{\partial U_z}{\partial z} + \frac{\partial p}{\partial z} \\ = \varepsilon\left(\frac{\partial^2 u_z}{\partial r^2} + \frac{1}{r}\frac{\partial u_z}{\partial r} - \frac{q^2}{r^2}u_z + \frac{\partial^2 u_z}{\partial z^2}\right) \end{aligned} \tag{A.9d}$$

In some cases, it may be preferable to express these equations in terms of the streamfunction because many base flows are two-dimensional or three dimensional but decoupled. If applicable, this has distinct advantages. The first being the reduced computational requirements for the resolution of less dependent variables. Related to the first, differentiating and summing the streamfunction equations can eliminate the pressure terms and reduce the system of four equations to a single equation [1, 123]. However, by keeping this system in general

form, we can construct a general algorithm that extends our capabilities to analyze base flows including weakly asymmetric properties and three-dimensional vector fields. A similar construction for the biglobal stability equations in Cartesian coordinates is discussed by Robitaillié-Montané and Casalis [124] and in Bipolar coordinates by Merzari *et al.* [125].

Appendix B

Numerical Codes

B.1 Polynomial Interpolation

This code generates Fig. 3.2. Its purpose is to illustrate the errors accrued by the Runge phenomenon; however, it works equally well in forming equispaced or Chebyshev polynomial approximations by changing the function f.

Algorithm B.1.1. *Equispaced and Chebyshev Interpolation (Matlab)*

input : The polynomial order N

output : Interpolating polynomials plotted against the original function

```
% Equis_vs_Cheb_Interpolation.m
% Find equispaced and Chebyshev polynomials
close all;
N = 16; i=1;
xx = -1.01:.005:1.01;
if i==1, x = -1 + 2*(0:N)/N; end                    % equispaced points
if i==2, x = cos(pi*(0:N)/N); end                    % Chebyshev points
f = 1./(1+16*x.^2);
p = polyfit(x,f,N);                                        % interpolation
pp = polyval(p,xx);                            % evaluation of interpolant
plot(x,f,'.','markersize',13)
line(xx,pp,'linewidth',.8)
figure
scatter(x,zeros(1,N+1))
```

Here we introduce two new functions, *polyfit* and *polyval*. The first determines the coefficients of the Nth order polynomial with the value f at the collocation points x. The second evaluates the polynomial at all points contained within xx. This function allows a smooth function to be plotted even with a coarse collocation grid. We will refrain from using this in further examples in favor of more stable methods. Algorithm B.4.1 employs barycentric interpolation, which if applied here, returns a stable result even for the equally spaced discretization but this defeats the purpose of the example.

B.2 Chebyshev Interpolating Polynomial Generator

This code will generate the Nth order Chebyshev polynomial defined by Eqs. (3.8–3.9) in the text and restated here as

$$\Pi_N f(\xi) = \sum_{i=0}^{N} f(\xi_i)\lambda_i(\xi) \tag{B.1}$$

and

$$\lambda_i(\xi) = (-1)^{i+1}\left(\frac{1-\xi^2}{\xi-\xi_i}\right)\left[\frac{T'_{N-1}(\theta)}{d_i N^2}\right] \tag{B.2}$$

where $\xi = \cos(i\pi/N)$ and $\theta = \arccos\xi$.

Please note that throughout the document we used the nomenclature N to denote the order of discretization. However, N is protected in Mathematica so we concede to using simply n.

Algorithm B.2.1. *Chebyshev Interpolating Polynomials (Mathematica)*

input : The order n of Chebyshev polynomial to be used

output : The nth order Chebyshev interpolating polynomial

(*Trefethen-Chebyshev Polynomial Generator.nb*)

(*Find Chebyshev Polynomials of order n*)

$n = 2;$

$\text{For}\,[j = 2, j < n - 1, j{+}{+}, d_j = 1]$

$d_1 = 2; d_n = 2;$

$\text{For}[j = 1, j < n + 1, j + +, \xi_j = \text{Cos}[(j + 1)\pi/(n - 1)]];$

$\theta = \text{ArcCos}[\xi];$

$\text{Tp}[n\_]{:=}(n - 1)\dfrac{\text{Sin}[(n - 1)\theta]}{\text{Sin}[\theta]};$

$\text{Simplify}\left[\sum_{i=1}^{n}(-1)^i \dfrac{1 - \xi^2}{\xi - \xi_i}\dfrac{\text{Tp}[n]}{d_i(n - 1)^2} f_i\right]$

B.3 Chebyshev Pseudo-Spectral Differentiation Matrix Generator

This section of code is the work of Weideman and Reddy [97] and can be downloaded at http://www.mathworks.com/matlabcentral/fileexchange/29. It is a powerful tool and utilized throughout this research. For completeness, it is listed here as well.

Algorithm B.3.1. *Chebyshev Pseudo-Spectral Differentiation Matrix (Matlab)*

input : The number of collocation points, N and the number of derivatives to compute, M

output : The $N \times N$, $1 - M^{\text{th}}$ order Chebyshev pseudo-spectral differentiation matrix

```
function [x, DM] = chebdif(N, M)

% The function [x, DM] = chebdif(N,M) computes the differentiation
% matrices D1, D2, ..., DM on Chebyshev nodes.
%
% Input:
% N:        Size of differentiation matrix.
% M:        Number of derivatives required (integer).
% Note:     0 < M ≤ N-1.
%
% Output:
% DM:       DM(1:N,1:N,ell) contains ell-th derivative matrix, ell=1..M.
%
% The code implements two strategies for enhanced
% accuracy suggested by W. Don and S. Solomonoff in
% SIAM J. Sci. Comp. Vol. 6, pp. 1253--1268 (1994).
% The two strategies are (a) the use of trigonometric
% identities to avoid the computation of differences
% x(k)-x(j) and (b) the use of the "flipping trick"
% which is necessary since sin t can be computed to high
% relative precision when t is small whereas sin (pi-t) cannot.
% Note added May 2003:  It may, in fact, be slightly better not to
% implement the strategies (a) and (b).   Please consult the following
% paper for details:   "Spectral Differencing with a Twist", by
% R. Baltensperger and M.R. Trummer, to appear in SIAM J. Sci. Comp.

% J.A.C. Weideman, S.C. Reddy 1998.  Help notes modified by
% JACW, May 2003.

    I = eye(N);                                          % Identity matrix.
    L = logical(I);                                 % Logical identity matrix.
   n1 = floor(N/2); n2  = ceil(N/2);     % Indices used for flipping trick.
    k = [0:N-1]';                                     % Compute theta vector.
   th = k*pi/(N-1);
    x = sin(pi*[N-1:-2:1-N]'/(2*(N-1)));        % Compute Chebyshev points.
    T = repmat(th/2,1,N);
   DX = 2*sin(T'+T).*sin(T'-T);                   % Trigonometric identity.
   DX = [DX(1:n1,:); -rot90(DX(1:n2,:),2)];                % Flipping trick.
DX(L) = ones(N,1);                       % Put 1's on the main diagonal of DX.
    C = toeplitz((-1).^k);                              % C is the matrix with
```

```
C(1,:) = C(1,:)*2; C(N,:) = C(N,:)*2;                    % entries c(k)/c(j)
C(:,1) = C(:,1)/2; C(:,N) = C(:,N)/2;
     Z = 1./DX;                              % Z contains entries 1/(x(k)-x(j))
  Z(L) = zeros(N,1);                              % with zeros on the diagonal.
     D = eye(N);                                   % D contains diff. matrices.
for ell = 1:M
          D = ell*Z.*(C.*repmat(diag(D),1,N) - D);          % Off-diagonals
       D(L) = -sum(D');                              % Correct main diagonal of D
DM(:,:,ell) = D;                                        % Store current D in DM
end
```

B.4 Chebyshev Interpolation

This section of code is the work of Weideman and Reddy [97] and can be downloaded at http://www.mathworks.com/matlabcentral/fileexchange/29. It is acts to calculate the polynomial interpolant from discretized equations.

Algorithm B.4.1. *Chebyshev Interpolation (Matlab)*

input : The collocation points, ξ, and the function values, f

output : The polynomial interpolant

```
function p = chebint(fk, x)

% The function p = chebint(fk, x) computes the polynomial interpolant
% of the data (xk, fk), where xk are the Chebyshev nodes.
% Two or more data points are assumed.
%
% Input:
% fk:  Vector of y-coordinates of data, at Chebyshev points
%      x(k) = cos((k-1)*pi/(N-1)), k = 1...N.
% x:   Vector of x-values where polynomial interpolant is to be evaluated.
%
% Output:
% p:    Vector of interpolated values.
%
```

```
% The code implements the barycentric formula; see page 252 in
% P. Henrici, Essentials of Numerical Analysis, Wiley, 1982.
% (Note that if some fk > 1/eps, with eps the machine epsilon,
% the value of eps in the code may have to be reduced.)

% J.A.C. Weideman, S.C. Reddy 1998

  fk = fk(:); x = x(:);                    % Make sure data are column vectors.
   N = length(fk);
   M = length(x);
  xk = sin(pi*(N-1:-2:1-N)'/(2*(N-1)));          % Compute Chebyshev points.
   w = ones(N,1).*(-1).^(0:N-1)';          % w = weights for Chebyshev formula
w(1) = w(1)/2; w(N) = w(N)/2;
   D = x(:,ones(1,N)) - xk(:,ones(1,M))';         % Compute quantities x-x(k)
   D = 1./(D+eps*(D==0));                            % and their reciprocals.
   p = D*(w.*fk)./(D*w);                            % Evaluate interpolant as
                                                    % matrix-vector products.
```

The key to effectively using this code is to realize that it is only useful if the input xk is defined between $[-1, 1]$ even though any other calculations can be done over any interval.

B.5 Computing a Spectral Derivative

This code calculates the spectral derivative of $\sin(x)$ from $[0, 2\pi]$.

Algorithm B.5.1. *Computing a Spectral Derivative*

input : The number of collocation points, N and the function, f to differentiate

output : The spectral derivative of function f

```
% App_of_Spec_Diff.m
% Finding a spectral derivative
close all;clc;clear
N=7;                                          % Define Number of Points
A=0;B=2*pi;                  % Define the left and right bounds, respectively
[xi,D]=chebdif(N,1);      % Compute Collocation points and Derivative Matrix
```

```
x=B/2*(xi+1)-A/2*(xi-1);                % Convert back to the original domain
d_dx=2/(B-A)*D(:,:,1);% Convert the Spectral derivatives back to the domain
f=d_dx*sin(x);
xx=linspace(0,2*pi);        % Define a finer grid spacing for smooth plotting
plot(xx,cos(xx),'b')
hold on
plot(xx,chebint(f,linspace(-1,1)),'r')                    % Plot the results
```

B.6 A First order ODE with Chebyshev Collocation

This code uses spectral methods to solve the first order ODE given by

$$f'(x) + \cos(x)f(x) = 0 \quad \text{with} \quad f(0) = 1 \tag{B.3}$$

Algorithm B.6.1. *Example: A First Order ODE with Chebyshev Collocation*

input : The number of collocation points, N, the spectral operator matrices, and boundary conditions

output : The solution to the ODE

```
% Cheb_1st_Order_ODE.m
% ODE Solver using Chebyshev Collocation Methods
close all;clc;clear
N=20;                                                 % Define Number of Points
A=0;B=pi;                     % Define the left and right bounds, respectively
a=1;                                        % Define the function at the bounds
[xi,D]=chebdif(N,1);       % Compute Collocation points and Derivative Matrix
x=B/2*(xi+1)-A/2*(xi-1);                % Convert back to the original domain
d_dx=2/(B-A)*D(:,:,1);% Convert the Spectral derivatives back to the domain
I=eye(N);                                           % Define the identity matrix
Bvec=zeros(N,1);                                   % Define the forcing function
Amat=d_dx+diag(cos(4*x))*I;                              % Define the matrix 'A'
Amat(N,:)=I(N,:);                                % Define the left B.C. operator
Bvec(N)=a;                                          % Define the left B.C. value
f=Amat\Bvec;                                          % Solve the equation Af=B
```

```
scatter(x,f,'.','r')                                    % Plot the spectral solution
hold on                                                 % Plot the exact solution
xx=linspace(A,B);          % Define a finer grid spacing for smooth plotting
F=exp(-1/4*sin(4*xx));
P=chebint(f,linspace(-1,1));
plot(xx,F,'b',xx,P,'r')                                 % Plot the results
[V,I]=max(F'-P);                          % Determine the Maximum Local Error
% ==================================================================
% Test for correct solution via numerically substituting into gov eq.
% ==================================================================
% Test the inner nodes
% ------------------------------------------------------------------
sub_test=d_dx*f+diag(cos(4*x))*f;i=2:N-1;
disp(['Back substitution results in ' num2str(max(max(abs(sub_test(i)))))])
% ------------------------------------------------------------------
% Test the boundary conditions
% ------------------------------------------------------------------
% Ax,Bx -->
Ax_check=Bvec(N)-f(N);
disp('Boundary condition check:')
disp(['Ax error ' num2str(max(abs(Ax_check)))])
% ==================================================================
% Test for correct solution via comparison with the Exact solution
% ==================================================================
disp(['Max Local Error: ', num2str(V), ' at ', num2str(xx(I))])
disp(['Error from Solving Af=B:  ' num2str(norm(Amat*f-Bvec))])
```

B.7 A Second Order BVP with Chebyshev Collocation

This code uses spectral methods to solve the BVP given by

$$f''(x) + xf'(x) + f(x) = x \quad \text{with} \quad \begin{cases} f(0) = 0 \\ f(4) = 1 \end{cases} \tag{B.4}$$

Algorithm B.7.1. *Example: A Second Order BVP with Chebyshev Collocation*

input : The number of collocation points, N, the spectral operator matrices, and boundary

conditions

output : The solution to the BVP

```
% Cheb_2nd_Order_ODE.m
% ODE Solver using Chebyshev Collocation Methods
clc;clf;clear;
N=20;                                                   % Define Number of Points
A=0;B=4;                        % Define the left and right bounds, respectively
a=0;b=1;                                      % Define the function at the bounds
[xi,D]=chebdif(N,1);        % Compute Collocation points and Derivative Matrix
x=B/2*(xi+1)-A/2*(xi-1);                    % Convert back to the original domain
d_dx=2/(B-A)*D(:,:,1);% Convert the Spectral derivatives back to the domain
d2_dx2=d_dx^2;
Q=diag(x);                                              % Define the coefficient
Bvec=x;                                               % Define the forcing function
I=eye(N);                                              % Define the identity matrix
Amat=d2_dx2+Q*d_dx+I;                                   % Define the matrix 'A'
Amat(1,:)=I(1,:);                                  % Define the right B.C. operator
Amat(N,:)=I(N,:);                                   % Define the left B.C. operator
Bvec(1)=b;                                            % Define the right B.C. value
Bvec(N)=a;                                             % Define the left B.C. value
f=Amat\Bvec;                                            % Solve the equation Af=B
scatter(x,f,'.','r')                                   % Plot the spectral solution
hold on                                                 % Plot the exact solution
xx=linspace(A,B);           % Define a finer grid spacing for smooth plotting
F=xx/2-mfun('dawson',xx/sqrt(2))/mfun('dawson',2*sqrt(2));
P=chebint(f,linspace(-1,1));
plot(xx,F,'b',xx,P,'r')                                         % Plot the results
[V,I]=max(F'-P);                                % Determine the Maximum Local Error
% ==================================================================================
% Test for correct solution via numerically substituting into gov eq.
% ==================================================================================
% Test the inner nodes
% ----------------------------------------------------------------------------------
sub_test=d2_dx2*f+Q*d_dx*f+f-Bvec;i=2:N-1;
disp(['Back substitution results in ' num2str(max(max(abs(sub_test(i)))))])
% ----------------------------------------------------------------------------------
% Test the boundary conditions
```

```
% ----------------------------------------------------------------------
% Ax,Bx -->
Ax_check=Bvec(N)-f(N);
Bx_check=Bvec(1)-f(1);
disp('Boundary condition check:')
disp(['Ax error ' num2str(max(abs(Ax_check))) ...
    '   Bx error ' num2str(max(abs(Bx_check)))])
% ======================================================================
% Test for correct solution via comparison with the Exact solution
% ======================================================================
disp(['Max Local Error: ', num2str(V), ' at ', num2str(xx(I))])
disp(['Error from Solving Af=B:  ' num2str(norm(Amat*f-Bvec))])
```

B.8 Systems of ODEs with Chebyshev Collocation

This code uses spectral methods to solve the system of ODEs

$$\begin{cases} f''(x) + xf'(x) - g(x) = \sin(4x) \\ g''(x) + g'(x) + f(x) = \cos(4x) \end{cases} \quad \text{with} \quad \begin{cases} f(0) = 1; \quad f'(0) = 10 \\ g(0) = 1; \quad g'(0) = 0 \end{cases} \tag{B.5}$$

Algorithm B.8.1. *Example: A System of ODEs with Chebyshev Collocation*

input : The number of collocation points, N, the spectral operator matrices, and boundary conditions

output : The solution to the System of ODES

```
% Systems_of_ODEs.m
% ODE Solver using Chebyshev Collocation Methods
clc;clf;clear;
N=20;                                          % Define Number of Points
i=1:1:N;j=1:1:N;            % Define iterators for simplicity of programming
A=0;B=10;                   % Define the left and right bounds, respectively
[xi,D]=chebdif(N,1);      % Compute Collocation points and Derivative Matrix
x=B/2*(xi+1)-A/2*(xi-1);                % Convert back to the original domain
d_dx=2/(B-A)*D(:,:,1);% Convert the Spectral derivatives back to the domain
```

```
d2_dx2=d_dx^2;
I=eye(N);                                   % Define the identity matrix
Bvec=zeros(2*N,1);                                % Allocate space for B
Amat=zeros(2*N,2*N);                              % Allocate space for A
% ---------------------------------------------------------------------
Bvec(i)=sin(4*x);                   % Define the forcing function from Eq.1
Bvec(N+i)=cos(4*x);                 % Define the forcing function from Eq.2
Amat(i,j)=d2_dx2+diag(x)*d_dx;         % Define the operator on f from Eq.1
Amat(i,N+j)=-I;                        % Define the operator on g from Eq.1
Amat(N+i,j)=I;                         % Define the operator on f from Eq.2
Amat(N+i,N+j)=d2_dx2+d_dx;             % Define the operator on g from Eq.2
Amat(1,j)=d_dx(N,:);              % Define the IV operator on f from BC on f
Amat(1,N+j)=0;                    % Define the IV operator on g from BC on f
Bvec(1)=10;                                          % Define the IV on f
Amat(N,j)=I(N,:);                 % Define the BC operator on f from BC on f
Amat(N,N+j)=0;                    % Define the BC operator on g from BC on f
Bvec(N)=1;                                           % Define the BC on f
Amat(N+1,j)=0;                    % Define the IV operator on f from BC on g
Amat(N+1,N+j)=d_dx(N,:);          % Define the IV operator on g from BC on g
Bvec(N+1)=0;                                         % Define the IV on g
Amat(2*N,j)=0;                    % Define the BC operator on f from BC on g
Amat(2*N,N+j)=I(N,:);             % Define the BC operator on g from BC on g
Bvec(2*N)=1;                                         % Define the BC on g
% ---------------------------------------------------------------------
f=Amat\Bvec;                                     % Solve the equation Af=B
% f=pinv(Amat)*Bvec;
scatter(x,f(i),'.','r')                   % Plot the spectral solution for u
hold on
scatter(x,f(N+i),'.','c')                 % Plot the spectral solution for y
xx=linspace(A,B);          % Define a finer grid spacing for smooth plotting
P1=chebint(f(1:N),linspace(-1,1));                      % Interpolate for u
P2=chebint(f(N+1:2*N),linspace(-1,1));                  % Interpolate for y
plot(xx,P1,'r',xx,P2,'c')                                % Plot the results
% =====================================================================
% Test for correct solution via numerically substituting into gov eq.
% =====================================================================
% Test the inner nodes
% ---------------------------------------------------------------------
sub_test1=d2_dx2*f(1:N)+diag(x)*d_dx*f(1:N)-f(N+1:2*N)-sin(4*x);
sub_test2=d2_dx2*f(N+1:2*N)+d_dx*f(N+1:2*N)+f(1:N)-cos(4*x);
```

```
i=2:N-1;
disp(['Back substitution results in '...
    num2str(max(max(abs(sub_test1(i)))))])
disp(['Back substitution results in '...
    num2str(max(max(abs(sub_test2(i)))))])
% ----------------------------------------------------------------------
% Test the boundary conditions
% ----------------------------------------------------------------------
% Ax,Bx -->
Ax_check=Bvec(1)-d_dx(N,:)*f(1:N);
Bx_check=Bvec(N)-f(N);
Ay_check=Bvec(N+1)-d_dx(N,:)*f(N+1:2*N);
By_check=Bvec(2*N)-f(2*N)';
disp('Boundary condition check:')
disp(['Ax error ' num2str(max(abs(Ax_check))) ...
    '   Bx error ' num2str(max(abs(Bx_check)))])
disp(['Ax error ' num2str(max(abs(Ay_check))) ...
    '   Bx error ' num2str(max(abs(By_check)))])
% ======================================================================
disp(['Error from Solving Af=B:  ' num2str(norm(Amat*f-Bvec))])
```

This system can still use MATLAB's built-in Gaussian elimination command $Amat\backslash Bvec$ to solve the system. For singular or nearly singular systems it becomes necessary to approximate the inverse with the command $pinv(Amat) * Bvec$. This command is slower and slightly less accurate, but may be the only option for some problems.

B.9 Eigenvalue Problems for ODEs

This code uses spectral methods to solve the eigenvalue problem

$$x^2 f''(x) + x f'(x) + \mu^2 x^2 f(x) = 0 \quad \text{with} \quad \begin{cases} f'(0) = 0 \\ f(10) = 0 \end{cases} \tag{B.6}$$

Algorithm B.9.1. *Example: Eigenvalue Problems for ODEs*

input : The number of collocation points, N, the spectral operator matrices

output : The eigenvalues and eigenvectors

```
% Cheb_Bessel_Eigs.m
% ODE Eigenvalue Problem Solver using Chebyshev Collocation Methods
% Bessel Equation
clear;clc;close all
N=20;                                                   % Define Number of Points
A=0;B=10;                      % Define the left and right bounds, respectively
a=0;b=1;                                      % Define the function at the bounds
[xi,D]=chebdif(N,1);        % Compute Collocation points and Derivative Matrix
x=B/2*(xi+1)-A/2*(xi-1);                    % Convert back to the original domain
d_dx=2/(B-A)*D(:,:,1);% Convert the Spectral derivatives back to the domain
d2_dx2=d_dx^2;
I=eye(N);                                           % Define the identity matrix
% ---------------------------------------------------------------------------
% Define the Generalize Eigenvalue Problem Af-lam*Bf=0
% ---------------------------------------------------------------------------
Amat=diag(x.^2)*d2_dx2+diag(x)*d_dx;                  % Define the matrix 'A'
Bmat=diag(x.^2);                                      % Define the matrix 'B'
Amat(1,:)=I(1,:);                     % Define the right B.C. in the 'A' matrix
Amat(N,:)=d_dx(N,:);                   % Define the left B.C. in the 'A' matrix
Bmat(1,:)=0;                          % Define the right B.C. in the 'B' matrix
Bmat(N,:)=0;                           % Define the left B.C. in the 'B' matrix
% ---------------------------------------------------------------------------
[V,Lam]=eig(Amat,-Bmat);
Lam = diag(Lam);Lam2=Lam;                % this forces +- infinty to the bottom
[foo,ii] = sort(abs(Lam2));                     % sort and index the eigenvalues
% ---------------------------------------------------------------------------
lam = sqrt(Lam(ii));             % Compute the eig of the original problem and
V = V(:,ii);                        % reorder the eigenvalues and eigenvectors
disp('The first 5 eigenvalues are')
format long;disp(lam(1:5));format short;
for j = 1:1:5          % Plot the eigenvectors against the BesselJ_0 solution
    if V(N,j)<0; V(:,j)=-V(:,j);end % Correct the signs of the eigenvectors
% ===========================================================================
% Test for correct solution via numerically substituting into gov eq.
```

```
% ==========================================================================
% Test the inner nodes
% --------------------------------------------------------------------------
    sub_test=diag(x.^2)*d2_dx2*V(:,j)+...
        diag(x)*d_dx*V(:,j)+lam(j)^2*diag(x.^2)*V(:,j);
    disp(['Back substitution results in ' ...
    num2str(max(max(abs(sub_test))))])
% --------------------------------------------------------------------------
% Test the boundary conditions
% --------------------------------------------------------------------------
    % Ax,Bx -->
    Dx=d_dx*V(:,j);
    Ax_check=0-Dx(N);
    Bx_check=0-V(1,j);
    % Bx,By -->
    disp('Boundary condition check:')
    disp(['Ax error ' num2str(max(abs(Ax_check))) ...
        '   Bx error ' num2str(max(abs(Bx_check)))])
% ==========================================================================
    plot(x,V(:,j),'.','markersize',12);hold on
    xx = linspace(A,B); P=chebint(V(:,j),linspace(-1,1));
    F=besselj(0,lam(j)*xx);
    plot(xx,F,'b',xx,P,'r')                                  % Plot the results
    [E,I]=max(F'-P);                        % Determine the Maximum Local Error
    disp(['Max Local Error: ', num2str(E), ' at ', num2str(xx(I))])
end
disp('Normalized Eigenvalues:')          % Display the normalized eigenvalues
format long;disp(lam(1:5)*(B-A));format short;
```

The eigenvectors V are the solution to the differential equation with the corresponding eigenvalues. In this case, it returns the Bessel function of the first kind.

B.10 A Parabolic Partial Differential Equation with Variable Coefficients

This code uses spectral methods to solve the PDE

$$f_y = \cos[y(1-x)]f_{xx} - xf \tag{B.7}$$

with boundary conditions

$$\begin{cases} f(x,0) = -2\cos(\pi x) \\ f_x(0,y) = 0 \\ f_x(1,y) = -2 \end{cases} \tag{B.8}$$

Algorithm B.10.1. *Example: Parabolic PDE w/ Variable Coefficients*

input : The number of collocation points, N, the spectral operator matrices, and boundary conditions

output : The two-dimensional solution

```
% Cheb_Heat_PDE_V3.m
% PDE Solver using Chebyshev Collocation Methods with nonzero BC's
% f_y-cos(y*(1-x))*f_xx+x*f=0
clc;clf;clear;
%% Set up the Grid and Independent Variables and their Derivatives:
% ==================================================================
N = 20;                 % Define the number of points in the x and y directions
Ax=0;Bx=1;                % Define the left and right bounds in x, respectively
Ay=0;By=1;                % Define the left and right bounds in y, respectively
[xi,D] = chebdif(N,1); % Define x collocation nodes and spectral derivative
   eta = xi;                                  % Define the y collocation points
x=Bx/2*(xi+1)-Ax/2*(xi-1);              % Convert back to the original domain
y=By/2*(eta+1)-Ay/2*(eta-1);            % Convert back to the original domain
d_dx=2/(Bx-Ax)*D(:,:,1);       % Convert the x derivatives back to the domain
d_dy=2/(By-Ay)*D(:,:,1);       % Convert the y derivatives back to the domain
d2_dx2=d_dx^2;                                  % Compute higher derivatives
```

```
d2_dy2=d_dy^2;
I = eye(N);                                          % Define the identity matrix
[xx,yy] = meshgrid(x,y);                                              % Mesh the grid
xx = xx(:); yy = yy(:);    % Convert the meshed grid to vectors to match the
                                        % tensor product form of the governing eq.
                                                  % and map the boundary conditions
%% Define the Operator Matrix
% ==========================================================================
II=eye(N^2);                        % Define an identity matrix the size of Amat
% Define the spectral operator matrix
Amat = kron(I,d_dy) -diag(cos((1-xx).*yy)) * kron(d2_dx2,I)+diag(xx)*II;
% --------------------------------------------------------------------------
% Impose boundary conditions by replacing appropriate elements
% --------------------------------------------------------------------------
Axbc = find(xx==Ax); Bxbc = find(xx==Bx);     % Find and store the locations
                                                      % of the x boundaries in
                                                    % order to find the points
                                                      % in the operator matrix
Aybc = find(yy==Ay); Bybc = find(yy==By);     % Find and store the locations
                                                      % of the y boundaries in
                                                    % order to find the points
                                                      % in the operator matrix
% --------------------------------------------------------------------------
% Define the differentiation matrix for boundary conditions
DDy=kron(I,d_dy);DDx=kron(d_dx,I);
% --------------------------------------------------------------------------
% Superimpose rows corresponding to the boundary conditions over the
% operator matrix.  The rows have been found and stored
% in Axbv Bxbc Aybc Bybc
% --------------------------------------------------------------------------
Amat(Axbc,:) = DDx(Axbc,:);
Amat(Bxbc,:) = DDx(Bxbc,:);
Amat(Aybc,:) = II(Aybc,:);
% Amat(Bybc,:) = II(Bybc,:);
% ==========================================================================
%% Define the RHS forcing function
% ==========================================================================
Bvec = zeros(N^2,1);            % Define the forcing function as the B vector
% --------------------------------------------------------------------------
% Define the value of the Boundary Conditions
```

```
% ---------------------------------------------------------------------
Bvec(Axbc) = 0;                                   % Define u(x,y) at x=Ax
Bvec(Bxbc) = -2;                                  % Define u(x,y) at x=Bx
Bvec(Aybc) = -2*cos(pi*xx(Aybc));                 % Define u(x,y) at y=Ay
% Bvec(Bybc) = 0;                                 % Define u(x,y) at y=By
% ====================================================================
% Solve the equation and reshape to 2D
% ====================================================================
f = Amat\Bvec;                                    % Solve the equation
% f=pinv(Amat)*Bvec;
ff = reshape(f,N,N);   % Convert the vector solution to x and y matrix form
% ====================================================================
% Test for correct solution via numerically substituting into gov eq.
% ====================================================================
% Test the inner nodes
% ---------------------------------------------------------------------
Dx=zeros(N,N);
Dy=zeros(N,N);
for i=1:N
    Dx(i,:)=(d2_dx2*ff(i,:)');
    Dy(:,i)=d_dy*ff(:,i);
end
yy=reshape(yy,N,N);xx=reshape(xx,N,N);          % Reshape back to matrix form
sub_test=Dy-cos(yy.*(1-xx)).*Dx+xx.*ff-reshape(Bvec,N,N);i=2:N-1;
disp(['Back substitution results in ' ...
    num2str(max(max(abs(sub_test(i,i)))))])
% ---------------------------------------------------------------------
% Test the boundary conditions
% ---------------------------------------------------------------------
% Ax,Bx -->
for i=1:N
    Dx(i,:)=(d_dx*ff(i,:)');
end
Ax_check=Bvec(Axbc)-Dx(:,N); % This is the first column
Bx_check=Bvec(Bxbc)-Dx(:,1);
% Bx,By -->
Ay_check=Bvec(Aybc)-ff(N,:)';
By_check=Bvec(Bybc)-ff(1,:)';
disp('Boundary condition check:')
disp(['Ax error ' num2str(max(abs(Ax_check(1:N-1)))) ...
```

```
    '   Bx error ' num2str(max(abs(Bx_check(1:N-1))))])
disp(['Ay error ' num2str(max(abs(Ay_check))) ])
% ==================================================================
% Set up finer grid, interpolate over fine grid, and plot
% ==================================================================
xgrid=2*N; ygrid=2*N;              % Define the number if interplation points
gridspacex=linspace(Ax,Bx,xgrid); % Define a finer grid spacing for x space
gridspacey=linspace(Ay,By,ygrid); % Define a finer grid spacing for y space

[xxx,yyy] = meshgrid(gridspacex,gridspacey);   % Define the finer grid mesh
P=zeros(N,xgrid);                % Preallocate the x interpolation polynomial
PP=zeros(ygrid,xgrid);           % Preallocate the y interpolation polynomial
% ------------------------------------------------------------------
% Iterpolate in the solution in the x direction
% ------------------------------------------------------------------
for i=1:N
    P(i,:)=chebint(ff(i,:),linspace(-1,1,xgrid));
end
% ------------------------------------------------------------------
% Iterpolate in the solution in the y direction
% ------------------------------------------------------------------
for j=1:xgrid
    PP(:,j)=chebint(P(:,j),linspace(-1,1,ygrid));
end
% ------------------------------------------------------------------
% Plot the solution
% ------------------------------------------------------------------
mesh(xxx,yyy,PP), colormap([0 0 0])
%  axis([0 1 0 1 0 1])
xlabel x, ylabel y, zlabel f
disp(['Error from Solving Af=B:  ' num2str(norm(Amat*f-Bvec))])
```

B.11 The Time-Independent Poisson Equation with a Sinusoidal Forcing Function

This code uses spectral methods to solve the PDE

$$f_{xx} + f_{yy} = 10 \sin\left[8x(y-1)\right], \qquad -1 < x, y < 1, \quad f = 0 \ \text{ at all boundaries} \tag{B.9}$$

Algorithm B.11.1. *Example: Poisson's Equation*

input : The number of collocation points, N, the spectral operator matrices, and boundary conditions

output : The two-dimensional solution

```
% Cheb_Poisson_PDE.m
% PDE Solver using Chebyshev Collocation Methods
% f_{xx}+f_{yy}=10*sin(8x(y-1))
%% Set up the Grid and Independent Variables and their Derivatives:
% ==================================================================
clc;clf;clear;
N = 20;              % Define the number of points in the x and y directions
Ax=-1;Bx=1;            % Define the left and right bounds in x, respectively
Ay=-1;By=1;            % Define the left and right bounds in y, respectively
[xi,D] = chebdif(N,1); % Define x collocation nodes and spectral derivative
   eta = xi;                                  % Define the y collocation points
x=Bx/2*(xi+1)-Ax/2*(xi-1);              % Convert back to the original domain
y=By/2*(eta+1)-Ay/2*(eta-1);              % Convert back to the original domain
d_dx=2/(Bx-Ax)*D(:,:,1);      % Convert the x derivatives back to the domain
d_dy=2/(By-Ay)*D(:,:,1);      % Convert the y derivatives back to the domain
d2_dx2=d_dx^2;                                  % Compute higher derivatives
d2_dy2=d_dy^2;
I = eye(N);                                     % Define the identity matrix
[xx,yy] = meshgrid(x,y);                                      % Mesh the grid
xx = xx(:); yy = yy(:);    % Convert the meshed grid to vectors to match the
                                   % tensor product form of the governing eq.
                                             % and map the boundary conditions
```

```
%% Define the Operator Matrix
% ==========================================================================
% Define the spectral operator matrix
Amat = kron(d2_dx2,I) + kron(I,d2_dy2);
% --------------------------------------------------------------------------
% Impose boundary conditions by replacing appropriate elements
% --------------------------------------------------------------------------
Axbc = find(xx==Ax); Bxbc = find(xx==Bx);  % Find and store the locations
                                           % of the x boundaries in
                                           % order to find the points
                                           % in the operator matrix
Aybc = find(yy==Ay); Bybc = find(yy==By);  % Find and store the locations
                                           % of the y boundaries in
                                           % order to find the points
                                           % in the operator matrix
% --------------------------------------------------------------------------
% Superimpose rows corresponding to the boundary conditions over the
% operator matrix.  The rows have been found and stored
% in Axbv Bxbc Aybc Bybc
% --------------------------------------------------------------------------
II=eye(N^2);                    % Define an identity matrix the size of Amat
% DDx=kron(I,d_dx);               % Define the larger differentiation matrix
% DDy=kron(d_dy,I);
Amat(Axbc,:) = II(Axbc,:);
Amat(Bxbc,:) = II(Bxbc,:);
Amat(Aybc,:) = II(Aybc,:);
Amat(Bybc,:) = II(Bybc,:);
%% Define the RHS forcing function
% ==========================================================================
Bvec = 10*sin(8*xx.*(yy-1));  % Define the forcing function as the B vector
% --------------------------------------------------------------------------
% Define the value of the Boundary Conditions
% --------------------------------------------------------------------------
Bvec(Axbc) = 0;                                  % Define f(x,y) at x=Ax
Bvec(Bxbc) = 0;                                  % Define f(x,y) at x=Bx
Bvec(Aybc) = 0;                                  % Define f(x,y) at y=Ay
Bvec(Bybc) = 0;                                  % Define f(x,y) at y=By
% ==========================================================================
% Solve the equation, reshape to 2D, and plot:
% ==========================================================================
```

```
f = Amat\Bvec;                                          % Solve the equation
% f = pinv(Amat)*Bvec;
ff = reshape(f,N,N);   % Convert the vector solution to x and y matrix form
% ==================================================================
% Test for correct solution via numerically substituting into gov eq.
% ==================================================================
% Test the inner nodes
% ------------------------------------------------------------------
Dx=zeros(N,N);
Dy=zeros(N,N);
for i=1:N
    Dx(i,:)=(d2_dx2*ff(i,:)');
    Dy(:,i)=d2_dy2*ff(:,i);
end
sub_test=Dy+Dx-reshape(Bvec,N,N);i=2:N-1;
disp(['Back substitution results in ' ...
    num2str(max(max(abs(sub_test(i,i)))))])
% ------------------------------------------------------------------
% Test the boundary conditions
% ------------------------------------------------------------------
% Ax,Bx -->
Ax_check=Bvec(Axbc)-ff(:,N);
Bx_check=Bvec(Bxbc)-ff(:,1);
% Bx,By -->
Ay_check=Bvec(Aybc)-ff(N,:)';
By_check=Bvec(Bybc)-ff(1,:)';
disp('Boundary condition check:')
disp(['Ax error ' num2str(max(abs(Ax_check))) ...
    '   Bx error ' num2str(max(abs(Bx_check)))])
disp(['Ay error ' num2str(max(abs(Ay_check))) ...
    '   By error ' num2str(max(abs(By_check)))])
% ==================================================================
% Set up finer grid, interpolate over fine grid, and plot
% ==================================================================
xgrid=2*N; ygrid=2*N;              % Define the number if interplation points
gridspacex=linspace(Ax,Bx,xgrid); % Define a finer grid spacing for x space
gridspacey=linspace(Ay,By,ygrid); % Define a finer grid spacing for y space
[xxx,yyy] = meshgrid(gridspacex,gridspacey);   % Define the finer grid mesh
P=zeros(N,xgrid);                % Preallocate the x interpolation polynomial
PP=zeros(ygrid,xgrid);           % Preallocate the y interpolation polynomial
```

```
% ----------------------------------------------------------------------
% Iterpolate in the solution in the x direction
% ----------------------------------------------------------------------
for i=1:N
    P(i,:)=chebint(ff(i,:),linspace(-1,1,xgrid));
end
% ----------------------------------------------------------------------
% Iterpolate in the solution in the y direction
% ----------------------------------------------------------------------
for j=1:xgrid
    PP(:,j)=chebint(P(:,j),linspace(-1,1,ygrid));
end
% ----------------------------------------------------------------------
% Plot the solution
% ----------------------------------------------------------------------
mesh(xxx,yyy,PP), colormap([0 0 0])
xlabel x, ylabel y, zlabel f
disp(['Error from Solving Af=B:  ' num2str(norm(Amat*f-Bvec))])
```

B.12 Systems of PDEs with Chebyshev Collocation

This code uses spectral methods to solve the system of PDEs

$$\begin{cases} f_{xx} + xf_{yy} - g_x = \sin(\pi x) \\ g_{xx} + g_{yy} + f_x = \cos(\pi x) \end{cases} \quad \text{with} \quad \begin{cases} f(0,y) = 0; & f(1,y) = 0 \\ f(x,0) = \sin(\pi x); & f(x,1) = 0 \\ g(0,y) = 0; & g(1,y) = 0 \\ g_y(x,0) = 0; & g(x,1) = 1 \end{cases} \tag{B.10}$$

Algorithm B.12.1. *Example: Systems of PDEs*

input : The number of collocation points, N, the spectral operator matrices, and boundary conditions

output : The two-dimensional solution

```
% Systems_of_PDEs.m
% PDE Solver using Chebyshev Collocation Methods with nonzero BC's
clc;close all;clear;
%% Set up the Grid and Independent Variables and their Derivatives:
% ==========================================================================
N = 20;              % Define the number of points in the x and y directions
Ax=0;Bx=1;             % Define the left and right bounds in x, respectively
Ay=0;By=1;             % Define the left and right bounds in y, respectively
[xi,D]=chebdif(N,1);% Define the collocation points and spectral derivative
   eta = xi;                                  % Define the y collocation points
x=Bx/2*(xi+1)-Ax/2*(xi-1);              % Convert back to the original domain
y=By/2*(eta+1)-Ay/2*(eta-1);            % Convert back to the original domain
d_dx=2/(Bx-Ax)*D(:,:,1);       % Convert the x derivatives back to the domain
d_dy=2/(By-Ay)*D(:,:,1);       % Convert the y derivatives back to the domain
d2_dx2=d_dx^2;                                  % Compute higher derivatives
d2_dy2=d_dy^2;
I = eye(N);                                     % Define the identity matrix
[xx,yy] = meshgrid(x,y);                                        % Mesh the grid
xx = xx(:); yy = yy(:);    % Convert the meshed grid to vectors to match the
                                        % tensor product form of the governing eq.
                                                % and map the boundary conditions
%% Define the Operator Matrix
% ==========================================================================
j=1:1:N^2;
% Define the operator on f from Eq.1
Amat(j,j) = kron(d2_dx2,I)+ diag(xx)*kron(I,d2_dy2);
% Define the operator on g from Eq.1
Amat(j,N^2+j) = -kron(d_dx,I);
% Define the operator on f from Eq.2
Amat(N^2+j,j) = kron(d_dx,I);
% Define the operator on g from Eq.2
Amat(N^2+j,N^2+j) = kron(d2_dx2,I)+ kron(I,d2_dy2);
%% Define Boundary Conditions
% ---------------------------------------------------------------------------
% Impose boundary conditions by replacing appropriate elements
% ---------------------------------------------------------------------------
Axbc = find(xx==Ax); Bxbc = find(xx==Bx);     % Find and store the locations
                                                        % of the x boundaries in
                                                       % order to find the points
                                                        % in the operator matrix
```

```
Aybc = find(yy==Ay); Bybc = find(yy==By);     % Find and store the locations
                                                  % of the y boundaries in
                                                 % order to find the points
                                                  % in the operator matrix
% ---------------------------------------------------------------------
% Superimpose rows corresponding boundary conditions over the operator
% matrix.  The rows have been found and stored in Axbv Bxbc Aybc Bybc
% ---------------------------------------------------------------------
II=eye(N^2);                    % Define an identity matrix the size of Amat
DDy=kron(I,d_dy);
DDx=kron(d_dx,I);                % Define the differentiation matrix for BCs
% ====================================================================
%% Define the RHS forcing function
% ====================================================================
Bvec(j,1)=sin(pi*xx);               % Define the forcing function from Eq.1
Bvec(N^2+j,1)=cos(pi*xx);           % Define the forcing function from Eq.2
% ---------------------------------------------------------------------
% Impose the 'f' boundary conditions on the block matrices representing the
% first equation
% ---------------------------------------------------------------------
% f(0,y) and f(1,y)
Amat(Axbc,j) = II(Axbc,:);        % Define the IV operator on f from BC on f
Amat(Axbc,N^2+j)=0*II(Axbc,:);    % Define the IV operator on g from BC on f
Bvec(Axbc) = 0;                                     % Define the IV on f
Amat(Bxbc,j) = II(Bxbc,:);        % Define the BC operator on f from BC on f
Amat(Bxbc,N^2+j)=0*II(Bxbc,:);    % Define the BC operator on g from BC on f
Bvec(Bxbc) = 0;                                     % Define the BC on f

% f(x,0) and f(x,1)
Amat(Aybc,j) = II(Aybc,:);        % Define the IV operator on f from BC on f
Amat(Aybc,N^2+j)=0*II(Aybc,:);    % Define the IV operator on g from BC on f
Bvec(Aybc) = sin(pi*xx(Aybc));                      % Define the IV on f
Amat(Bybc,j) = II(Bybc,:);        % Define the BC operator on f from BC on f
Amat(Bybc,N^2+j)=0*II(Bybc,:);    % Define the BC operator on g from BC on f
Bvec(Bybc) = sin(pi*xx(Bybc));                      % Define the BC on f

% g(0,y) and g(1,y)
Amat(N^2+Axbc,j)=0*II(Axbc,:);    % Define the IV operator on f from BC on g
Amat(N^2+Axbc,N^2+j)=II(Axbc,:); % Define the IV operator on g from BC on g
Bvec(N^2+Axbc)=0;                                   % Define the IV on g
```

```
Amat(N^2+Bxbc,j)=0*II(Bxbc,:);    % Define the BC operator on f from BC on g
Amat(N^2+Bxbc,N^2+j)=II(Bxbc,:); % Define the BC operator on g from BC on g
Bvec(N^2+Bxbc)=0;                                          % Define the BC on g

% g(x,0) and g(x,1)
Amat(N^2+Aybc,j)=0*DDy(Aybc,:);  % Define the IV operator on f from BC on g
Amat(N^2+Aybc,N^2+j)=DDy(Aybc,:);% Define the IV operator on g from BC on g
Bvec(N^2+Aybc)=0;                                          % Define the IV on g
Amat(N^2+Bybc,j)=0*II(Bybc,:);    % Define the BC operator on f from BC on g
Amat(N^2+Bybc,N^2+j)=II(Bybc,:); % Define the BC operator on g from BC on g
Bvec(N^2+Bybc)=0;                                          % Define the BC on g

% ==========================================================================
% Solve the equation and reshape to 2D
% ==========================================================================
f = Amat\Bvec;                                             % Solve the equation
% f=pinv(Amat)*Bvec;
ff = reshape(f(j),N,N);% Convert the vector solution to x and y matrix form
gg = reshape(f(N^2+j),N,N);
% ==========================================================================
% Test for correct solution via numerically substituting into gov eq.
% ==========================================================================
% Test the inner nodes
% --------------------------------------------------------------------------
Dxf=zeros(N,N);
Dyf=zeros(N,N);
Dxxf=zeros(N,N);
Dyyf=zeros(N,N);

Dxg=zeros(N,N);
Dyg=zeros(N,N);
Dxxg=zeros(N,N);
Dyyg=zeros(N,N);

for i=1:N
    Dxxf(i,:)=(d2_dx2*ff(i,:)');
    Dxf(i,:)=(d_dx*ff(i,:)');
    Dyyf(:,i)=d2_dy2*ff(:,i);
    Dyf(:,i)=d_dy*ff(:,i);
```

```
    Dxxg(i,:)=(d2_dx2*gg(i,:)');
    Dxg(i,:)=(d_dx*gg(i,:)');
    Dyyg(:,i)=d2_dy2*gg(:,i);
    Dyg(:,i)=d_dy*gg(:,i);
end

yy=reshape(yy,N,N);xx=reshape(xx,N,N);          % Reshape back to matrix form
sub_test1=Dxxf+xx.*Dyyf-Dxg-reshape(Bvec(j),N,N);
sub_test2=Dxxg+Dyyg+Dxf-reshape(Bvec(N^2+j),N,N);i=2:N-1;
disp(['Back substitution results in ' ...
    num2str(max(max(abs(sub_test1(i,i)))))])
disp(['Back substitution results in ' ...
    num2str(max(max(abs(sub_test2(i,i)))))])
% ----------------------------------------------------------------------
% Test the boundary conditions
% ----------------------------------------------------------------------
% BCs on f
Ax_check=Bvec(Axbc)-ff(:,N);
Bx_check=Bvec(Bxbc)-ff(:,1);
Ay_check=Bvec(Aybc)-ff(N,:)';
By_check=Bvec(Bybc)-ff(1,:)';

disp('Boundary condition check for f(x,y):')
disp(['Ax error ' num2str(max(abs(Ax_check))) ...
    '   Bx error ' num2str(max(abs(Bx_check)))])
disp(['Ay error ' num2str(max(abs(Ay_check))) ...
    '   By error ' num2str(max(abs(By_check)))])
% ----------------------------------------------------------------------
% BCs on g
Ax_check=Bvec(N^2+Axbc)-gg(:,N);
Bx_check=Bvec(N^2+Bxbc)-gg(:,1);
Ay_check=Bvec(N^2+Aybc)-Dyg(N,:)';
By_check=Bvec(N^2+Bybc)-gg(1,:)';

disp('Boundary condition check for g(x,y):')
disp(['Ax error ' num2str(max(abs(Ax_check(1:N-1)))) ...
    '   Bx error ' num2str(max(abs(Bx_check)))])
disp(['Ay error ' num2str(max(abs(Ay_check))) ...
    '   By error ' num2str(max(abs(By_check)))])
% ======================================================================
```

```
% Set up finer grid, interpolate over fine grid, and plot
% ==================================================================
xgrid=2*N; ygrid=2*N;              % Define the number if interplation points
gridspacex=linspace(Ax,Bx,xgrid);   % Define a finer grid spacing for the x
gridspacey=linspace(Ay,By,ygrid);   % Define a finer grid spacing for the y

[xxx,yyy] = meshgrid(gridspacex,gridspacey);   % Define the finer grid mesh
Pf=zeros(N,xgrid);              % Preallocate the x interpolation polynomial
PPf=zeros(ygrid,xgrid);         % Preallocate the y interpolation polynomial
Pg=zeros(N,xgrid);              % Preallocate the x interpolation polynomial
PPg=zeros(ygrid,xgrid);         % Preallocate the y interpolation polynomial

% ------------------------------------------------------------------
% Interpolate in the solution in the x direction
% ------------------------------------------------------------------
for i=1:N
    Pf(i,:)=chebint(ff(i,:),linspace(-1,1,xgrid));
    Pg(i,:)=chebint(gg(i,:),linspace(-1,1,xgrid));
end
% ------------------------------------------------------------------
% Interpolate in the solution in the y direction
% ------------------------------------------------------------------
for j=1:xgrid
    PPf(:,j)=chebint(Pf(:,j),linspace(-1,1,ygrid));
    PPg(:,j)=chebint(Pg(:,j),linspace(-1,1,ygrid));
end
% ------------------------------------------------------------------
% Plot the solution
% ------------------------------------------------------------------
figure
mesh(xxx,yyy,PPf), colormap([0 0 0])
xlabel x, ylabel y, zlabel f
figure
mesh(xxx,yyy,PPg), colormap([0 0 0])
xlabel x, ylabel y, zlabel g
disp(['Error from Solving Af=B:  ' num2str(norm(Amat*f-Bvec))])
```

B.13 Eigenvalue Problems for PDEs

This code uses spectral methods to solve the Helmholtz equation

$$f_{xx} + f_{yy} + \mu f = 0, \quad 0 \le x, y \le L, \quad f = 0 \text{ on all boundaries} \tag{B.11}$$

Algorithm B.13.1. *Example: Eigenvalue Problems for PDEs*

input : The number of collocation points, N, the spectral operator matrices

output : The eigenvalues and eigenvectors

```
% Cheb_PDE_Eigs.m
% Set up tensor product Laplacian and compute 4 eigenmodes:
% f_xx+f_yy+\lambda*f=0
clc;clf;clear;
% ==================================================================
N = 30;              % Define the number of points in the x and y directions
Ax=0;Bx=1;             % Define the left and right bounds in x, respectively
Ay=0;By=1;             % Define the left and right bounds in y, respectively
[xi,D] = chebdif(N,1); % Define x collocation nodes and spectral derivative
   eta = xi;                                  % Define the y collocation points
x=Bx/2*(xi+1)-Ax/2*(xi-1);              % Convert back to the original domain
y=By/2*(eta+1)-Ay/2*(eta-1);             % Convert back to the original domain
d_dx=2/(Bx-Ax)*D(:,:,1);      % Convert the x derivatives back to the domain
d_dy=2/(By-Ay)*D(:,:,1);      % Convert the y derivatives back to the domain
d2_dx2=d_dx^2;                                  % Compute higher derivatives
d2_dy2=d_dy^2;
I = eye(N);                                     % Define the identity matrix
[xx,yy] = meshgrid(x,y);                                      % Mesh the grid
xx = xx(:); yy = yy(:);   % Convert the meshed grid to vectors to match the
                                  % tensor product form of the governing eq.
II=eye(N^2);                      % Define an identity matrix the size of Amat
Amat = kron(d2_dx2,I) + kron(I,d2_dy2);                          % Laplacian
% Amat=kron(d2_dx2,I)+kron(I,d2_dy2)-diag(exp(20*(yy-xx-1)));%+perturbation
Bmat = II;
Cvec=zeros(N^2,1);
% ------------------------------------------------------------------
```

```
% Impose boundary conditions by replacing appropriate elements
% ----------------------------------------------------------------------
Axbc = find(xx==Ax); Bxbc = find(xx==Bx);     % Find and store the locations
                                                  % of the x boundaries in
                                                % order to find the points
                                                  % in the operator matrix
Aybc = find(yy==Ay); Bybc = find(yy==By);     % Find and store the locations
                                                  % of the y boundaries in
                                                % order to find the points
                                                  % in the operator matrix
% ----------------------------------------------------------------------
% Superimpose rows corresponding to the boundary conditions over the
% operator matrix.  The rows have been found and stored
% in Axbv Bxbc Aybc Bybc
% ----------------------------------------------------------------------
II=eye(N^2);                      % Define an identity matrix the size of Amat
DDx=kron(I,d_dx);                   % Define the larger differentiation matrix
DDy=kron(d_dy,I);

Amat(Axbc,:) = II(Axbc,:);
Amat(Bxbc,:) = II(Bxbc,:);
Amat(Aybc,:) = II(Aybc,:);
Amat(Bybc,:) = II(Bybc,:);

Bmat(Axbc,:) = 0*II(Axbc,:);
Bmat(Bxbc,:) = 0*II(Bxbc,:);
Bmat(Aybc,:) = 0*II(Bybc,:);
Bmat(Bybc,:) = 0*II(Bybc,:);

Cvec(Axbc,:) = 0;
Cvec(Bxbc,:) = 0;
Cvec(Aybc,:) = 0;
Cvec(Bybc,:) = 0;
% ----------------------------------------------------------------------
[V,Lam] = eig(Amat,-Bmat);
Lam = diag(Lam);Lam2=Lam;             % this forces +- infinty to the bottom
[foo,ii] = sort(abs(Lam2));
% ----------------------------------------------------------------------
lam=Lam(ii);
V = V(:,ii);
```

```
disp('The first 5 eigenvalues are')
format long;disp(lam(1:5));format short;
% ==========================================================================
% Set up finer grid, interpolate over fine grid, and plot
% ==========================================================================
xgrid=2*N; ygrid=2*N;              % Define the number if interplation points
gridspacex=linspace(Ax,Bx,xgrid); % Define a finer grid spacing for x space
gridspacey=linspace(Ay,By,ygrid); % Define a finer grid spacing for y space
[xxx,yyy] = meshgrid(gridspacex,gridspacey);   % Define the finer grid mesh
P=zeros(N,xgrid);                % Preallocate the x interpolation polynomial
PP=zeros(ygrid,xgrid);           % Preallocate the y interpolation polynomial
xx = reshape(xx,N,N); yy = reshape(yy,N,N);
[ay,ax] = meshgrid([.56 .04],[.1 .5]);

for k = 1:4
    ff = reshape(V(:,k),N,N);
% Test for correct solution via numerically substituting into gov eq.
% ==========================================================================
% Test the inner nodes
% --------------------------------------------------------------------------
    Dx=zeros(N,N);
    Dy=zeros(N,N);
    for i=1:N
        Dx(i,:)=(d2_dx2*ff(i,:)');
        Dy(:,i)=d2_dy2*ff(:,i);
    end
    sub_test=Dy+Dx+lam(k)*ff;
%     sub_test=Dy+Dx+(lam(k)-exp(20*(yy-xx-1))).*ff;
        disp(['Back substitution results in ' ...
        num2str(max(max(abs(sub_test(2:N-1,2:N-1)))))])
% --------------------------------------------------------------------------
% Test the boundary conditions
% --------------------------------------------------------------------------
    % Ax,Bx -->
    Ax_check=0-ff(:,end);
    Bx_check=0-ff(:,1);
    % Bx,By -->
    Ay_check=0-ff(end,:);
    By_check=0-ff(1,:);
    disp('Boundary condition check:')
```

```
    disp(['Ax error ' num2str(max(abs(Ax_check))) ...
        '   Bx error ' num2str(max(abs(Bx_check)))])
    disp(['Ay error ' num2str(max(abs(Ay_check))) ...
        '   By error ' num2str(max(abs(By_check)))])
% ----------------------------------------------------------------------
% Iterpolate in the solution in the x direction
% ----------------------------------------------------------------------
    for i=1:N
        P(i,:)=chebint(ff(i,:),linspace(-1,1,xgrid));
    end
% ----------------------------------------------------------------------
% Iterpolate in the solution in the y direction
% ----------------------------------------------------------------------
    for j=1:xgrid
        PP(:,j)=chebint(P(:,j),linspace(-1,1,ygrid));
    end
    subplot('position',[ax(k) ay(k) .38 .38])
%     contour(xxx,yyy,PP,-.9:.2:.9), colormap([0 0 0]), axis square
    mesh(xxx,yyy,PP)%, colormap([0 0 0])
    title(['eig = ' num2str(lam(k)/(pi^2/((Bx-Ax)*(By-Ay))),'%18.12f')...
        '\pi^2/L^2'])
end
disp('Normalized Eigenvalues:')        % Display the normalized eigenvalues
format long;disp(lam(1:5)*((Bx-Ax)*(By-Ay)));format short;

disp('For the table:')        % Display the normalized eigenvalues
format long;disp(lam(1:20)/(pi^2/((Bx-Ax)*(By-Ay))));format short;
```

B.14 Single Matrix Balancing

This code balances the norm of a single matrix following the procedure discussed by Wilkinson and Reinsch [112].

Algorithm B.14.1. *Single Matrix Balancing*

input : Matrix A_{ij}

output : The similar, balanced matrix A_{ij}

```
%% Matrix Balancing
% ==========================================================================

% This algorithm 'balances' nonsymmetric matrices.  This procedure is of
% order N^2 operations.  The time taken is small in comparison to other
% algorithms involved in calculating eigenvalues.  It is recommended to
% ALWAYS balance nonsymmetric matrices.  It doesn't hurt and can
% significantly inprove the accuracy in some cases.

% "The errors in the eigensystem found by a numerical procedure are
% generally proportional to the Euclidean norm of the matrix, that is,
% to the square root of the sum of the squares of the elements.  The idea
% of balancing is to use similarity transformations to make corresponding
% rows and columns of the matrix have comparible norms, thus reducing the
% overall norm of the matrix while leaving the eigenvalues unchanged."

% -from Numerical Recipes in C: The Art of Scientific Computing.

% This method works works for any matrix, real or complex.  If being
% transfered to another language a method for calculating the abs() of a
% complex number must be written.  |x+i*y|=sqrt(x^2+y^2)=modulus

clear;close all;clc;

%% Define Matrix
% ==========================================================================
A=rand(5);                                        % Generate a random matrix
disp(eigs(A))                                 % Display the original eigenvalues
%% Initialize Parameters
% ==========================================================================
RADIX=2; % the base or radix is usually the number of unique digits,
% including zero, that a positional numeral system uses to represent
% numbers. For example, for the decimal system (the most common system
% in use today) the radix is 10, because it uses the 10 digits from
% 0 through 9.  To avoid rounding errors, exact powers of the radix base
% are used for floating point arithmetic.

sqrdx=RADIX*RADIX;
last=0;
[m n]=size(A);
```

```
%% Begin Algorithm
% ==========================================================================
while last==0
    last=1;
    for I=1:1:n
        r=0;c=0;

% Calculate the 'taxicab' norms of the matrix ||x||=sum(|x|)
% ==========================================================================
        for J=1:1:n
            if J≠I
                c=c+abs(A(J,I));          % abs() is the absolute modulus for
                                                                % complex numbers
                r=r+abs(A(I,J));                      % same goes for this one
            end
        end
        if c≠0 && r≠0 % if both are nonzero
            g=r/RADIX;
            f=1;
            s=c+r;

% Find the integer power of the machine radix that is closest to balancing
% the matrix
% ==========================================================================
            while c<g
                f=f*RADIX;
                c=c*sqrdx;
            end
            g=r*RADIX;
            while c>g
                f=f/RADIX;
                c=c/sqrdx;
            end
            if (c+r)/f<0.95*s
                last=0;
                g=1/f;

% Apply similarity transformation
% ==========================================================================
```

```
                    for J=1:1:n
                        A(I,J)=A(I,J)*g;
                    end
                    for J=1:1:n
                        A(J,I)=A(J,I)*f;
                    end
% ==================================================================
                end
            end
        end
end
disp(eigs(A))                    % Verify the eigenvalues have not changed
```

B.15 Matrix Pencil Balancing

This code balances the norm of a matrix pencil following the procedure by Lemonnier and Van Dooren [111].

Algorithm B.15.1. *Matrix Pencil Balancing*

input : Matrix pencil matrices A_{ij} and B_{ij}

output : The similar, balanced matrix pencil matrices A_{ij} and B_{ij}

```
function [A B] = Generalized_Balance(A,B)
% Lemonnier and Van Dooren
% Performs two-sided scaling DL\A*DR, DL\B*DR in order to improve
% the sensitivity of generalized eigenvalues. The diagonal matrices
% DL and DR are constrained to powers of 2 and are computed iteratively
% until the number of iterations max_iter is met or until the norms are
% between 1/2 and 2. Convergence is often reached after 2 or 3 steps.
% The diagonals of the scaling matrices are returned in DL and DR
% and so is iter, the number of iterations steps used by the method.
% clc;
% clear;
% %% Define the matrix
% % ==================================================================
```

```
% d=.2; N=7;            % density and matrix size for sparce matrix generation
% A=full(sprand(N,N,d));A(2,5)=10000;     % A, modified to have poor balancing
% B=full(sprand(N,N,d));B(6,4)=150;       % B, modified to have poor balancing
% disp('Frobenius Norm');disp(norm(A,'fro'));disp(norm(B,'fro'))
% eig(A,B)                                                  % original eigenvalues
%% Define Parameters
% ==================================================================================
    max_iter=20;
    N=size(A,1);                                  % define the size of matrix A
    change=zeros(2,N);
    DL=ones(1,N); DR=ones(1,N);             % initialize the diagonals of the
                                                              % similarity matrix
    M=abs(A).^2+abs(B).^2;
%% Begin Formation of DL and DR
% ==================================================================================
    for iter=1:max_iter,
        emax=0;emin=0;
        for I=1:N;
                        % scale the rows of M to have approximate row sum 1
            d=sum(M(I,:));
            e=-round(log2(abs(d))/2);
            M(I,:)=pow2(M(I,:),2*e);
                        % apply the square root scaling also to A, B and DL
            DL(I)=pow2(DL(I),-e);
           % this is needed to avoid singularity in DL with sparse matrices
% ----------------------------------------------------------------------------
            if DL(I)==inf || DL(I)==0
                DL(I)=2;
            end
% ----------------------------------------------------------------------------
            if e > emax, emax=e; end;
            if e < emin, emin=e; end
        end
        for I=1:N;
                  % scale the columns of M to have approximate column sum 1
            d=sum(M(:,I));e=-round(log2(abs(d))/2);
            M(:,I)=pow2(M(:,I),2*e);
                        % apply the square root scaling also to A, B and DR
            DR(I)=pow2(DR(I),e);
                  % this is needed to avoid singularity in DL with sparse
```

```
                                                                    % matrices
% ---------------------------------------------------------------------
            if DR(I)==inf || DR(I)==0
                DR(I)=1;
            end
% ---------------------------------------------------------------------
            if e > emax, emax=e; end;
            if e < emin, emin=e; end
        end
            if (emax≤emin+2), break; end
                                % Stop if norms are all between 1/2 and 2
            if iter>1          % Stop if values of DL or DR are not changing
                change(1,1:N)=DL;change(2,1:N)=DR;
                TEST(1,1:N)=abs(change(1,1:N)-DL);
                TEST(2,1:N)=abs(change(2,1:N)-DR);
                if any(TEST)==0
                    break;
                end
                change(1,1:N)=DL;change(2,1:N)=DR;
            end
    end
% ======================================================================
    DR=diag(DR,0);                                  % form the matrix DR
    DLI=diag(1./DL,0);                                   % calc DL^(-1)
    A=DLI*A*DR;B=DLI*B*DR;
% ======================================================================
    disp('Frobenius Norm');disp(norm(A,'fro'));disp(norm(B,'fro'))
%       eig(A,B)
end
```

The code

```
% A=rand(10);A(2,5)=100;
% B=rand(10);B(6,4)=150;
A=full(sprand(10,10,.1));A(2,5)=1000;
B=full(sprand(10,10,.1));B(6,4)=150;
```

defines a random matrix of size 10×10. The function *rand* creates a random dense matrix, while *sprand* creates a sparse matrix of nonzero element density of 0.1. The higher the

density number, the less dense the matrix. Since the generalized eigenvalue problem can exhibit sparsity, it is important to code for both. Both commands generate matrices for which balancing is unnecessary. The explicitly defined elements $A(2,5)$ and $B(6,4)$ are large and therefore cause imbalance and large norms.

The definition $M = \text{abs}(A)^2 + \text{abs}(B)^2$ is in tune with calculating the norm. The variable d handles the rest. All calculations are done in M rather than on A and B.

The code snippets

```
if DL(I)==inf || DL(I)==0
    DL(I)=2;
end
% ------------------------------------------------------------------
if DR(I)==inf || DR(I)==0
    DR(I)=1;
end
```

are required to ensure nonsingular behavior for sparse matrices. From time to time, a zero element will appear on the diagonal. Since the inverse of a diagonal matrix (the variable DLI in the code) is 1./diag (i.e. $1/a_{1,1}, 1/a_{2,2}, \cdots 1/a_{n,n}$), a zero on the diagonal will cause singularity. The code checks for this and replaces it by a small positive number. The selection of the small positive number is somewhat arbitrary. If we adhere to the powers of the radix, then 2 is a good candidate. The selection of 1 for DR rather than 2 comes from observing that for dense matrices, a value of 1 is common and will minimize the norm. Mathematically, there is little basis for these choices. It is at the discretion of the programmer to select the best criteria to reduce the norm. The similarity transformation, as defined, is not directly determined by the diagonal matrices. As long as the desired norm reduction is achieved, any diagonal matrix is sufficient.

Convergence is determined when all the norms are between 1/2 and 2. This usually takes only two or three iterations. A second stopping criterion occurs if the values of DL and DR no longer change but this circumstance is rare.

One can verify the effectiveness of this code by calculating the norms before and after convergence. To further verify, the MATLAB command $eig(A,'nobalance')$ can be used. This command only works with a single matrix so we must define $B_{ij} = I_N$ (remember to remove the user defined, large value from B_{ij}. We add the following code before and after the transformations are applied.

```
disp('Frobenius Norm');disp(norm(A,'fro'));
eig(A,'nobalance') % ONLY BEFORE THE TRANSFORMATIONS ARE APPLIED
eig(A,B)
```

The first line displays the Frobenius norms of A before balancing is applied. The second line calculates the eigenvalues of A_{ij} without balancing, therefore computing the eigenvalues of a poorly conditioned matrix. The third line calculates the balanced eigenvalues of A_{ij} (remember $B_{ij} = I_N$).

After the transformations are applied $B_{ij} \neq I_N$ but the eigenvalues are still unchanged. One can see the significant norm reduction and agreement with the original call, $eig(A, B)$.

B.16 Segregating Nonzero Elements

This algorithm pushes rows and columns containing nonzero elements up and to the left, respectively, and forces as many nonzero diagonal elements as possible. This increases stability in the eigensolver and should be implemented before matrix reductions.

Algorithm B.16.1. *Segregating Nonzero Elements*

input : Matrix pencil matrices A_{ij} and B_{ij}

output : The similar, balanced matrix pencil matrices A_{ij} and B_{ij}

```
%% B Matrix Modification
% ==========================================================================
% This function determines if there is a zero on the diagonal.  If there
% is, it tests the other elements above it in the same column to see if the
% rows can be swapped to make a nonzero diagonal element.  If no suitable
% element is found, it shifts one column to the left and tries again.  If
% an element is found, it is moved from its original location to the
% diagonal position in question by row and column swapping.
function [A B]=B_Mod(A,B,N)

% Hessenberg transformation can still put zeros on the diagonals
% Zero Column/row test
% This forces any columns or rows with no nonzero elements to the left and
% top.  This should help with division by zero problems in the transforms
% function.
for I=N:-1:2
    if any(B(:,I))==false                                          % Column test
        for J=I-1:-1:1
            if any(B(:,J))==true
                Y=A(:,I);A(:,I)=A(:,J);A(:,J)=Y;
                Y=B(:,I);B(:,I)=B(:,J);B(:,J)=Y;
                break;
            end
        end
    end
    if any(B(I,:))==false                                          % Row test
        for J=I-1:-1:1
            if any(B(J,:))==true
                Y=A(I,:);A(I,:)=A(J,:);A(J,:)=Y;
                Y=B(I,:);B(I,:)=B(J,:);B(J,:)=Y;
                break;
            end
        end
    end
end

for I=N:-1:2
    FLAG=0;
    if B(I,I)==0
        for K=I:-1:1
```

```
            for J=I:-1:1
                if B(J,K)≠0
                    Y=A(I,:);A(I,:)=A(J,:);A(J,:)=Y;
                    Y=B(I,:);B(I,:)=B(J,:);B(J,:)=Y;
                    if K≠I
                        Y=A(:,I);A(:,I)=A(:,K);A(:,K)=Y;
                        Y=B(:,I);B(:,I)=B(:,K);B(:,K)=Y;
                    end
                    FLAG=1;
                    break;
                end
            end
            if FLAG==1;
                break;
            end
        end
    end
end


end
```

B.17 Real Symmetric Matrix to Tridiagonal Form

This algorithm converts a real symmetric matrix to real symmetric tridiagonal form using Householder transformations.

Algorithm B.17.1. *Real Symmetric to Tridiagonal*

input : Symmetric matrix A_{ij}

output : Symmetric tridiagonal matrix A_{ij}

```
%% HOUSEHOLDER'S METHOD FOR SYMMETRIC MATRICES
%==========================================================================
% See "Householder.pdf" for full algorithm
```

```
% Householder's method is used for several purposes.  One being
% orthogonalization and one being a method for reducing an arbitrary
% SYMMETRIC matrix to a similar tridiagonal matrix.  This is a necessary
% form for many eigensolvers

% This method allow us to change a REAL, SYMMETRIC n x n matrix A=a(i,j)
% into a tridiagonal matrix with the same set of eigenvalues.
% Let v be a column
% vector with ||v||_2=1 ( ||v||_2 = sqrt(|v1|^2+|v2|^2+...+) basically the
% magnitude of the vector).  The Householder transformation corresponding
% to the vector "v" is the orthogonal matrix
%                    $$H=I_n-2vv'$$

% Convention for variables
%          a(x,y):    x=row number
%                     y=column number

% REMINDER: The vectors are displayed horizontally in the matrix (input and
% output)

% profile on
%% DEFINE THE INPUT MATRIX
%==========================================================================
clear;close all;clc;tic

A = [1 -1 2 2;-1 2 1 -1;2 1 3 2;2 -1 2 1];
% A=sprandsym(10,10)
%% INITIALIZE PARAMETERS
%==========================================================================
tol=10^(-10);   % tolerance to determine what is numerically zero
[m n]=size(A);  % automatically defines the stopping criteria for the loop
v=zeros(1,n);   % initializes the vector
k=1;
%% BEGIN CALCULATIONS
%==========================================================================
while k<n-1
    s=sqrt(sum(A(k+1:n,k).^2));
    if s==0
        k=k+1;
        s=sqrt(sum(A(k+1:n,k).^2));
```

```
    end
    if A(k+1,k)<0
        SG=-1;
    else
        SG=1;
    end
    z=1/2*(1+SG*A(k+1,k)/s);
    for i=1:k
        v(i)=0;
    end
    v(k+1)=sqrt(z);
    for i=k+2:n
        v(i)=SG*A(k,i)/(2*v(k+1)*s);
    end
    v=v';
    H=eye(n)-2*(v)*v';                          % Calculating H=I_n-2v'v
    v=v';
    A=H*A*H;                                    % Defining the updated A
    k=k+1;
end

%  IDENTIFYING NUMERICAL ZERO
%==================================================================================
% This is an optional section that identifies values that are numerically
% zero and explicitly sets them to zero.  This might be important since
% those values COULD introduce sign errors.  I don't think it should matter
% though.  For large matrices this could be slow and unnecessary.  On
% average this section of code takes about 0.065 sec for a 100x100 matrix

for i=1:n
    for j=1:n
        if abs(A(i,j))≤tol
        A(i,j)=0;
        end
    end
end
disp(A)
toc
% profile viewer
```

B.18 Real Nonsymmetric Matrix to Upper Hessenberg Form

This algorithm converts a real nonsymmetric matrix to real upper Hessenberg form using Householder transformations.

Algorithm B.18.1. *Real Nonsymmetric to Upper Hessenberg*

input : Real nonsymmetric matrix A_{ij}

output : Real upper Hessenberg matrix A_{ij}

```
%% HOUSEHOLDER'S METHOD FOR ARBITRARY REAL* MATRICES
%==========================================================================
% See Burden and Faires "Numerical Analysis" for full algorithm.  This code
% is optimized for matlab

% Householder's method is used for several purposes.  One being
% orthogonalization and one being a method for reducing an arbitrary
% matrix to an UPPER HESSENBERG matrix.  This is a necessary
% form for many eigensolvers.  If the original matrix is symmetric
% Householder's Method will produce a SIMILAR TRIDIAGONAL MATRIX.

% This method allow us to change an arbitrary n x n matrix A=a(i,j) into
% an UPPER HESSENBERG matrix with the same set of eigenvalues.  Let w be a
% column vector with ||w||_2=1 ( ||w||_2 = sqrt(|w1|^2+|w2|^2+...+)
% basically the magnitude of the vector).
% The Householder transformation corresponding to the vector "w" is the
% orthogonal matrix
%                    P=I - 2ww'

% Convention for variables
%          a(x,y):    x=row number
%                     y=column number

% REMINDER: The vectors are displayed horizontally in the matrix (input and
% output).  Be sure to transpose the matrix if this code is being used as a
% function and vectors are passes vertically!!!!  It's done this way for
```

```
% simplicity of input only! Just be CAREFUL!

% For some reason running this on a upper Hessenberg Matrix returns
% opposite signs for some elements

% * It works for complex matrices as long as the elements below the lower
% subdiagonal are REAL!!!  For a completely general solution, this needs
% modified

close all;clc;clear
tic
% profile on
%% DEFINE THE INPUT MATRIX
%==========================================================================
A=rand(10);
disp(A)
eigs(A)
%% INITIALIZE PARAMETERS
%==========================================================================
tol=10^(-10);   % tolerance to determine what is numerically zero
[m n]=size(A);  % automatically defines the stopping criteria for the loop
v=zeros(1,n);
%% BEGIN CALCULATIONS
%==========================================================================
for k=1:n-2

% CALCULATE ALPHA & RSQ
%==========================================================================
    q=0;
    for j=k+1:n
        q=q+A(j,k)^2;
    end
    if A(k+1,k)==0
        alpha=-sqrt(q);
    else
        alpha=-(sqrt(q)*A(k+1,k))/abs(A(k+1,k));
    end

    RSQ=alpha^2-alpha*A(k+1,k);
```

```
% CALCULATE THE VECTORS
%==========================================================================
    v(k)=0;v(k+1)=A(k+1,k)-alpha;
    v(k+2:n)=A(k+2:n,k);

    u=1/RSQ*A*v';
    y=1/RSQ*v*A;

    PROD=v*u;

    z=u-1/RSQ*PROD*v';

% CALCULATE THE MATRIX
%==========================================================================
    for L=k+1:n
        A(L,1:k)=A(L,1:k)-y(1:k)*v(L);
        A(1:k,L)=A(1:k,L)-z(1:k)*v(L);
        A(k+1:n,L)=A(k+1:n,L)-z(k+1:n)*v(L)-y(L)*v(k+1:n)';
    end

end

%  IDENTIFYING NUMERICAL ZERO
%==========================================================================
% This is an optional section that identifies values that are numerically
% zero and explicitly sets them to zero.  This might be important since
% those values COULD introduce sign errors.  I don't think it should matter
% though.  For large matrices this could be slow and unnecessary.  On
% average this section of code takes about 0.065 sec for a 100x100 matrix

for i=1:m
    for j=1:n
        if abs(A(i,j))≤tol
        A(i,j)=0;
        end
    end
end

eigs(A)
% disp(A);
```

```
% The following does the same thing as the previous loops using built in
% matlab commands.  On average it takes 0.078 for a 100x100 matrix
% ---------------------------------------------------------------------
% r=abs(A)≤tol;
% A(r)=0;
% ---------------------------------------------------------------------
toc
```

B.19 Real/Complex Nonsymmetric Matrix to Upper Hessenberg Form

This algorithm converts a real or complex nonsymmetric matrix to real upper Hessenberg form using eliminations with pivoting.

Algorithm B.19.1. *Real/Complex Nonsymmetric to Upper Hessenberg*

input : Real/complex nonsymmetric matrix A_{ij}

output : Real/complex upper Hessenberg matrix A_{ij}

```
%% Reduction of a Real or Complex Nonsymmetric Matrix to Hessenberg Form
% ======================================================================

% This works for every matrix i've tested. Real, Complex, balanced,
% unbalanced, upper hessenberg, symmetric.  All of them without any errors.

% see Wilkinson's Handbook for Automatic Computations Vol 2: Linear Algebra
% and Numerical Recipes in C: The Art of Scientific Computing for
% methodology

% Stabilized elementary similarity transformations are used for reducing an
% unsymmetric complex matrix to an UPPER HESSENBERG matrix.  This is a
% necessary form for many eigensolvers. If the original matrix is symmetric
% this Method will produce a SIMILAR TRIDIAGONAL MATRIX.

% This procedure is similar to Gaussian elimination with pivoting.  It is
```

```
% about a factor of 2 more efficient that Householder's Method (see other
% codes).  This is actually a non-orthogonal method and a matrix can be
% constructed that this strategy is not stable and the orthogonal
% Householder's method is.  The matrices are extremely rare in practice so
% this method can be used with confidence.

% Gaussian elimination is not a similarity transformation so this is
% slightly different.  Before the rth stage, the original matrix A=A1
% becomes Ar (upper Hessenberg) in its first r-1 rows and columns.  The rth
% stage consists of the following operations:
% 1) Find the element of maximum magnitude in the rth column below the
% diagonal. If it is zero, skip the next two bullets and the stage is done
% Otherwise, suppose the maximum element was in row r'.
% 2) Interchange rows r' and r + 1. This is the pivoting procedure. To make
% the permutation a similarity transformation, also interchange columns r'
% and r + 1.
% 3) For i = r + 2, r + 3, . . .,N, compute the multiplier
% n(i,r+1) = a(i,r)/a(r+1,r)
% Subtract n(i,r+1) times row r + 1 from row i. To make the elimination a
% similarity transformation, also add n(i,r+1) times column i to column r+1

% This method allow us to change an arbitrary real or complex
% n x n matrix A=a(i,j) into an UPPER HESSENBERG matrix with the same set
% of eigenvalues. Because of the potentially complex nature of the original
% matrix this algorithm uses stabilized elementary similarity
% transformation
% Convention for variables
%         a(x,y):   x=row number
%                   y=column number

% REMINDER: The vectors are displayed horizontally in the matrix (input and
% output).  Be sure to transpose the matrix if this code is being used as a
% function and vectors are passes vertically!!!!  It's done this way for
% simplicity of input only! Just be CAREFUL!

close all;clc;clear
tic
% profile on
%% DEFINE THE INPUT MATRIX
%=================================================================
```

```
A=rand(5)+1i*rand(5);
% disp(A)
 eigs(A)

%% INITIALIZE PARAMETERS
%==========================================================================
[m n]=size(A);  % automatically defines the stopping criteria for the loop
interchange=zeros(1,n);
% The following parameters are output from a 'balancing' procedure to
% prepare 'A'.  If that procedure is not used, k=1 and L=n.

%% BEGIN CALCULATIONS
%==========================================================================
for m=2:1:n
%       A=Ar+i*Ai;
%       disp(A)
    I=m;x=0;

% Find the pivot
% ========================================================================
    for J=m:1:n
        if abs(A(J,m-1))>abs(x)
            x=A(J,m-1);
            I=J;
        end
    end
    interchange(m)=I;

% Interchange rows and columns of arrays 'Ar' and 'Ai'
% ========================================================================
        if I≠m
            for J=m-1:1:n
                y=A(I,J); A(I,J)=A(m,J); A(m,J)=y;
            end
            for J=1:1:n
                y=A(J,I); A(J,I)=A(J,m); A(J,m)=y;
            end
        end

% Carry out elimination
```

```
% ==================================================================
        if x≠0
            for I=m+1:1:n
                y=A(I,m-1);
                 if y≠0
                    y=y/x;
                      % Complex division is handled automatically in Matlab
                    A(I,:)=A(I,:)-y*A(m,:);
                    A(:,m)=A(:,m)+y*A(:,I);
                end
            end
        end

end

% Zero out elements below the lower subdiagonal
% ==================================================================
% This method leaves artifacts below the lower subdiagonal that
% are numerically zero.  The actual upper Hessenberg matrix is
% preserved and these values should be thought of as zero.  The following
% code 'zeros' out these elements.
for C=1:n
    for R=C+2:n
        A(R,C)=0;
    end
end
disp(A)
eigs(A)
```

B.20 Matrix Pencil Reduction

This algorithm converts a real or complex nonsymmetric matrix to real upper Hessenberg form using eliminations with pivoting.

Algorithm B.20.1. *Matrix Pencil Reduction*

input : Matrix pencil matrices A_{ij} and B_{ij}

output : Similar upper Hessenberg matrix A_{ij} and upper triangular matrix B_{ij}

```
%% Matrix Reductions
% ==========================================================================

% This function simultaneously reduces matrix A to upper Hessenberg and
% matrix B to triangular form and outputs the new A, B, and the vector X
% containing the transformations used in the reductions in order to
% calculate the eigenvectors.

% INPUT VARIABLES
% --------------------------------------------------------------------------
% N     =   size of A and B
% A,B   =   nXn complex matrix

% --------------------------------------------------------------------------
% OUTPUT VARIABLES
% --------------------------------------------------------------------------
% A     =   complex upper Hessenberg matrix - the original is destroyed
% B     =   complex upper triangular matrix - the original is destroyed
function [A B N] = LZ_HESS(A,B,N)

% Reduce B to Triangular form using Elementary transformations
% ==========================================================================
for I=1:N-1
% Find the Pivot
% --------------------------------------------------------------------------
    PIVOT=I;D=0; % an arbitrary value to initialize the loop
    for K=I+1:N
        if abs(B(K,I))>abs(D)
            D=B(K,I);
            PIVOT=K; % PIVOT is the pivot row index
        end
    end
% --------------------------------------------------------------------------
% Do we have to interchange rows? YES if PIVOT=¬I
% --------------------------------------------------------------------------
        if PIVOT≠I
                Y=A(I,:);    A(I,:)=A(PIVOT,:);      A(PIVOT,:)=Y;
                Y=B(I,:);    B(I,:)=B(PIVOT,:);      B(PIVOT,:)=Y;
        end
% --------------------------------------------------------------------------
```

```
% Apply Elimination Transformations
% ----------------------------------------------------------------------
        if D≠0
            for J=I+1:N
                Y=B(J,I)/B(I,I);
                if Y≠0
                    A(J,:)=A(J,:)-Y*A(I,:);
                    B(J,:)=B(J,:)-Y*B(I,:);
                end
            end
        end
        B(I+1:N,I)=0;
end
% ======================================================================
% Reduce A to upper Hessenberg form
% ----------------------------------------------------------------------
if N-2≥1
    for J=1:N-2
        for PIVOT=1:N-1-J
            I=N+1-PIVOT;
% ----------------------------------------------------------------------
% Do we have to interchange rows? YES if abs(A(I,J))>abs(A(I-1,J))
% ----------------------------------------------------------------------
            if abs(A(I,J))>abs(A(I-1,J))
                Y=A(I,:);    A(I,:)=A(I-1,:);    A(I-1,:)=Y;
                Y=B(I,:);    B(I,:)=B(I-1,:);    B(I-1,:)=Y;
            end
% ----------------------------------------------------------------------
% Carry out elimination
% ----------------------------------------------------------------------
            if A(I,J)≠0
                Y=A(I,J)/A(I-1,J);
                A(I,:)=A(I,:)-Y*A(I-1,:);
                B(I,:)=B(I,:)-Y*B(I-1,:);
            end
% ----------------------------------------------------------------------
% Do we have to interchange columns? YES if abs(B(I,I-1))>abs(B(I,I))
% ----------------------------------------------------------------------
            if abs(B(I,I-1))>abs(B(I,I))
                Y=B(:,I);    B(:,I)=B(:,I-1);    B(:,I-1)=Y;
```

```
                Y=A(:,I);    A(:,I)=A(:,I-1);    A(:,I-1)=Y;
            end
% ---------------------------------------------------------------------
% Carry out elimination to return B to upper triangular
% ---------------------------------------------------------------------
            if B(I,I-1)≠0
                Y=B(I,I-1)/B(I,I);
                B(:,I-1)=B(:,I-1)-Y*B(:,I);
                B(I,I-1)=0;
                A(:,I-1)=A(:,I-1)-Y*A(:,I);
            end
        end
        A(J+2:N,J)=0;
    end
end
% sort(eig(A,B))
end
```

B.21 Block Decomposition

This algorithm pushes rows and columns containing nonzero elements up and to the left, respectively, and forces as many nonzero diagonal elements as possible. This increases stability in the eigensolver and should be implemented before matrix reductions.

Algorithm B.21.1. *Block Decomposition*

input : An upper Hessenberg matrix A_{ij}, an upper triangular matrix B_{ij}, and the positions of the adjoining small subdiagonal elements

output : The original and decomposed eigenvalues

```
% BLOCK DECOMPOSITION
% ===================================================================
% This code verifies block decomposition when there exists a small
% subdiagonal element
clc; close all; clear;
```

```
A=hess(rand(10));B=triu(rand(10));
[M N]=size(A);
J=4;
A(J+1,J)=10^(-8);
A(J+2,J+1)=10^(-8);
ORIG=eig(A,B);
disp('Original Eigenvalues')
disp(ORIG)
% DIAG=diag(A)./diag(B);
% disp('diag(A)./diag(B)')
% disp(DIAG)
% disp('Difference between the two')
% disp(ORIG-DIAG)

G=A(1:J,1:J);H=B(1:J,1:J);
UPPER=eig(G,H);
disp('The Eigs of the Upper Block')
disp(UPPER)

Q=A(J+1:N,J+1:N);R=B(J+1:N,J+1:N);
LOWER=eig(Q,R);
disp('The Eigs of the Lower Block')
disp(LOWER)

EIG(1:J)=UPPER;
EIG(J+1:N)=LOWER;
disp('The original Eigs are')
disp(ORIG)
disp('The Block decomposed Eigs are')
disp(EIG')
disp('The difference is')
format long
disp(sort(ORIG)-sort(EIG'))
format short
```

B.22 The Power Method

This algorithm calculates the largest eigenvalue using deflation techniques.

Algorithm B.22.1. *The Power Method*

input : Matrix A_{ij} and a guess eigenvector x

output : The largest eigenvalue and its associated eigenvector

```
%% THE POWER METHOD
% ==================================================================
close all;clc;clear
% This m-file uses the Power Method to determine the largest magnitude
% eigenvalue of the n x n matrix A given a nonzero vector x.
% Aitken's procedure is used to speed up convergence

% The system must possess a eigenvalue that is distinctly larger than the
% others such that |L1|>|L2|>...|Ln|

% One useful feature of the Power Method is that it produces both the
% eigenvalue and the eigenvector.  This method can be applied to a known
% set of predetermined eigenvalues to determine their eigenvectors.

% REMINDER: The vectors are displayed horizontally in the matrix. Be sure
% to transpose the matrix if this code is being used as a
% function and vectors are passes vertically!!!!  It's done this way for
% simplicity of input only!

% The initial vector 'x' must be stated as the transpose
%% DEFINE THE INPUT MATRIX 'A' & INITIALIZATION VECTOR 'x'
% ==================================================================
A=rand(4);
n=size(A,1);
eig(A)
x=ones(n,1);                                    % "Guess" vector x(1,:)

%% INITIALIZE PARAMETERS
% ==================================================================
tol=10^(-6);                                    % convergence tolerance
N=2000;                                                 % max iterations

k=1;
```

```
%% THE SOLUTION
% ==========================================================================
while k≤N
% This allows any vector of proper size to be the initialization vector
% ==========================================================================
    [val, p]=max(abs(x));  % this determines the position of p and saves it
    x=x/x(p);              % this normalizes the vector

% This is the iteration on 'x'
% ==========================================================================
    x=(A*x);

% This renormalizes the new vector
% ==========================================================================
    [val, p]=max(abs(x));  % this determines the position of p and saves it
    if x(p)==0
        disp('Eigenvector')
        disp(x')
        disp('"A" has the eigenvalue 0, select a new vector x and restart')
        break;
    end
    x=x/x(p);                                  % this normalizes the vector

% This calculates the eigenvalue 'lamda' via the Rayleigh Quotient
% ==========================================================================
    lamda=(x'*A*x)/(x'*x);

% This tests the error and ends the calculations
% ==========================================================================
    if max(abs(A*x-(x*lamda)))<tol*abs(lamda)
        disp('The procedure was successful')
        disp('Eigenvalue')
        disp(lamda)
        disp('Eigenvector')
        disp(x')
        break;
    end
    k=k+1;
    if k==N+1
        disp('Max iterations exceeded')
```

```
        disp('The procedure failed')
        break;
    end

end
```

B.23 The Inverse Power Method

This algorithm calculates the closest eigenvalue to an initial guess, q, using deflation techniques.

Algorithm B.23.1. *The Inverse Power Method*

input : Matrix A_{ij} and a guess eigenvector x

output : The largest eigenvalue and its associated eigenvector

```
%% THE INVERSE POWER METHOD
% ==================================================================
close all;clc;clear
% This m-file uses the Inverse Power Method to determine the largest
% magnitude eigenvalue of the n x n matrix A given a nonzero vector x.
% Aitken's procedure is used to speed up convergence

% The system must possess a eigenvalue that is distinctly larger than the
% others such that |L1|>|L2|>...|Ln|

% One useful feature of the Power Method is that it produces both the
% eigenvalue and the eigenvector.  This method can be applied to a known
% set of predetermined eigenvalues to determine their eigenvectors.

% The Inverse Power Method is used to approximate an eigenvalue closest to
% a specified guess value.  It is often used when a general idea of the
% eigenvalue is already known.

% REMINDER: The vectors are displayed horizontally in the matrix (input and
% output).  Be sure to transpose the matrix if this code is being used as a
```

```
% function and vectors are passes vertically!!!!  It's done this way for
% simplicity of input only!

% The initial vector 'x' must be stated as the transpose

% tic
% profile on
%% DEFINE THE INPUT MATRIX 'A' & INITIALIZATION VECTOR 'x'
% ==================================================================
A=rand(4);
n=size(A,1);
x=ones(n,1);                                        % "Guess" vector x(1,:)
[V,D]=eig(A)
%% INITIALIZE PARAMETERS
% ==================================================================
tol=10^(-10);                                       % convergence tolerance
N=2000;                                                   % max iterations
[m n]=size(A);                          % automatically computes the size of A

k=1;
FLAG=0;
%% STEP 1:  DETERMINE A GUESS EIGENVALUE FROM THE INITIAL VECTOR
% ==================================================================
q=(x'*A*x)/(x'*x);       % this determines an appropriate guess eigenvalue
% the algorithm converges to the eigenvalue closest to 'q'.  It may not be
% the largest.  A user defined quantity of 'q' can be entered here to get a
% different eigenvalue.

%% THE SOLUTION
% ==================================================================
while k<=N
% This allows any vector of proper size to be the initialization vector
% ==================================================================
    [val, p]=max(abs(x));  % this determines the position of p and saves it
    x=x/x(p);                                   % this normalizes the vector

% This is the iteration on 'x'
% ==================================================================
    x=((A-q*eye(n))\x);
    if k==1              % this tests if q is an eigenvalue
```

```
        for i=1:n
            if isnan(x(i))==1 || isinf(x(i))==1
                disp('"q" is an eigenvalue')
                disp('The eigenvalue is')
                disp(q)
                disp('The eigenvector is')
                disp(x)
                FLAG=1;
                disp...
       ('Restarting with a perturbation on q to calculate the eigenvector')
                q=q+.01;
                x=ones(n,1);
                break;
            end

        end
    end
% This renormalizes the new vector
% ==================================================================
    [val, p]=max(abs(x));  % this determines the position of p and saves it
    if x(p)==0
        disp('Eigenvector')
        disp(x)
        disp('"A" has the eigenvalue 0, select a new vector x and restart')
        break;
    end
    x=x/x(p);                                   % this normalizes the vector

% This calculates the eigenvalue 'lamda' via the Rayleigh Quotient
% ==================================================================
    lamda=(x'*A*x)/(x'*x);

% This tests the error and ends the calculations
% ==================================================================
    if max(abs(A*x-(x*lamda)))<tol*abs(lamda)
        disp('The procedure was successful')
        disp('Eigenvalue')
        disp(lamda)
        disp('Eigenvector')
        disp(x)
```

```
        break;
    end
    k=k+1;
    if k==N+1
        disp('Max iterations exceeded')
        disp('The procedure failed')
        break;
    end

end
```

B.24 The QR Method

This algorithm calculates the spectrum of the real upper Hessenberg matrix A_{ij} using QR decompositions.

Algorithm B.24.1. *The QR Method*

input : Matrix A_{ij}

output : The spectrum

```
%% QR Algorithm for Real Upper Hessenberg Matrices
% ==================================================================

% This code calculates the eigenvalues of a real upper Hessenberg Matrix.
% Unlike the simplier symmetric version of this code, the original matrix
% must be of the upper Hessenberg.  This form will likely return complex
% eigenvalues.  Because of this, the following code is significantly
% different.  If the original matrix is not Upper Hessenberg, then the
% matrix must first be conditioned by Householder_Arbitrary.m.

% This code is iterative, but without a general loop.  The iteration is
% performed when conditional statements point the information to different
% subfunctions.  Ultimately, the code will finish and break the loop.

% This algorithm follows Wilkinson's text 'Handbook for Automatic
```

```
% Computation Vol 2 Linear Algebra

% To automate this code in conjunction with Householder_Arbitrary.m simply
% input the upper Hessenberg matrix from Householder_Arbitrary.m 'A' as the
% input matrix 'A' in this program.  Matrix 'A' is preserved in the
% program.

% !if there is a problem passing variables remember that not all the
% functions are passing the same variables.

function []=Real_Upper_Hessenberg_QR()
% Initialize the Program
%==========================================================================
clear;close all;clc;
tic

%% Define the Matrix
%==========================================================================

% The original matrix is defined and preserved as 'A'
J=5;
A=rand(J);
B=eye(J);
[A,B]=LZ_HESS(A,B,J);
% Define 'H' as 'A' to preserve 'A'.  All calculations are done on 'H'
H=A;
% eigs(H)

% Initialize Parameters
%==========================================================================
t=0;  % This is the 'exceptional shift' parameter for redundant shift loops
na=0;                                % This is a place holder related to 'n'
its=0;     % This is the iteration counter.  It determines the necessity to
                                                  % perform an exceptional shift.
[m n]=size(H);                % This automatically determines the size of 'H'
lambda=zeros(1,n);                                  % These are the eigenvalues

% These are arbitrary assignments that are a placeholder for variable
% passing in the first iteration.  They are rewritten with quickly.
p=0;q=0;r=0;s=0;w=0;x=0;y=0;z=0;L=0;
```

```
% Begin the Solution
% ==========================================================================
nextw(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);

end

%% Continuation Subfunction
% ==========================================================================

% This subfunction tests if all the eigenvalues have been solved.  If so it
% breaks the loop and displays the results.  If not it restarts the
% calculation to determine the next eigenvalue.
function [n H lambda p q r s t w x y z L m na its]...
    =nextw(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its)

if n==0

    disp('Calculation Successful')
    disp('The Eigenvalues are')
    format long % To increase the displayed digits
    disp(lambda')
    format short % To return the output digits back to normal
    toc
else
    its=0; na=n-1;
    [n H lambda p q r s t w x y z L m na its]...
        =nextit(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);
end

end

%% Begin the next iteration
% ==========================================================================
function [n H lambda p q r s t w x y z L m na its]...
    =nextit(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its)

if n≠0

% Look for a single small subdiagonal element
```

```
% ==========================================================================
% If a single small subdiagonal element is found, then the matrix is
% reduced to calculate the eigenvalues of the submatrix below the small
% subdiagonal.  This accelerates the convergence of the algorithm

for L=n:-1:2
    if abs(H(L,L-1))≤eps*(abs(H(L-1,L-1))+abs(H(L,L)))
        [n H lambda p q r s t w x y z m na its]...
            =cont1(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);
    end
end

% If no single small subdiagonal element is found, then 'L' is reset to '1'
% and the calculations continue as normal, without being able to take
% advantage of a reduced submatrix.

L=1;
[n H lambda p q r s t w x y z m na its]...
    =cont1(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);

% the subfunction called in either case is 'cont1'.  The difference is the
% value of 'L' which determines where the calculation will begin from; i.e.
% the entire matrix or a submatrix.
end

end

%% Continue 1:  Testing for Convergence
% ==========================================================================

% This subfunction begins by testing if the eigenvalues are explicitly
% calculated because either the original matrix or a submatrix has be
% completely reduced and the eigenvalue appears explicitly on the diagonal
% elements.  If an eigenvalue is not determined, then the either an
% exceptional shift is performed to avoid redundant shifting or two
% consecutive subdiagonal elements are tested for.  If two are found, they
% constitute the same affect as a single small subdiagonal.  This
% subfunction continues until the possible acceleration techniques are
% exhausted.  At this point, it continues on as normal.
```

```
function [n H lambda p q r s t w x y z m na its]...
    =cont1(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its)

if n≠0
x=H(n,n);

% Determine if a single eigenvalue has been found
% ==================================================================
if L==n
    [n H lambda p q r s t w x y z L m na its]...
        =onew(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);
end

% Completion test
% ==================================================================
% Since this code doesn't have a controling loop to end the computation, it
% will give out of bounds errors after the code successfully calculates the
% last eigenvalue.  The following statement avoids the error by cancelling
% the call to any following statements if the computation is complete.  The
% error happens to occur here first.  It's really an error of poor
% organization in the program.  oops.  Another one shows up in a few lines.
%  They work with the 'if n≠0' statements at the top of each function.
if n==0 || na==0
    return
end

% Determine if two eigenvalues have been found
% ==================================================================
y=H(na,na);
w=H(n,na)*H(na,n);
if L==na
    [n H lambda p q r s t w x y z L m na its]...
        =twow(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);
end

if n==0 || na==0
    return
end
% This tests for failure with the current exceptional shift applied
% ==================================================================
```

```
% Did not converge
if its==30
    disp('FAIL! Max iterations exceeded')
end
% Did not converge with current shift
if its==10 || its==20
    t=t+x;  % Form exceptional shift
    for i=1:1:n
        H(i,i)=H(i,i)-x;
    end
    s=abs(H(n,na))+abs(H(na,n-2));
    x=.75*s;
    y=x;
    w=-.4375*s^2;
end
its=its+1;  % Increase counter

% Look for two consecutive small subdiagonal elements
% ==========================================================================
for m=n-2:-1:L
    z=H(m,m);
    r=x-z;
    s=y-z;
    p=(r*s-w)/H(m+1,m)+H(m,m+1);
    q=H(m+1,m+1)-z-r-s;
    r=H(m+2,m+1);
    s=abs(p)+abs(q)+abs(r);
    p=p/s; q=q/s; r=r/s;
    if m==L || abs(H(m,m-1))*(abs(q)+abs(r))≤...
            eps*abs(p)*(abs(H(m-1,m-1))+abs(z)+abs(H(m+1,m+1)))
        [n H lambda p q r s t w x y z L na its]...
            =cont2(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);
    end
end

end
end

%% One Eigenvalue Found
% ==========================================================================
```

```
% If one eigenvalue is found, it must be a real number

function [n H lambda p q r s t w x y z L m na its]...
    =onew(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its)

if n≠0

lambda(n)=x+t; % The eigenvalue is determined from the exceptional shift
n=na;   % The last row and column is eliminated by this statement

% After a successful calculation, the procedure is restarted
[n H lambda p q r s t w x y z L m na its]...
    =nextw(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);

end
end

%% Two Eigenvalues are Found
% ==================================================================

% In the case of two Eigenvalues being found simultaneously they can be
% real or complex.

function [n H lambda p q r s t w x y z L m na its]...
    =twow(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its)

if n≠0
p=(y-x)/2; q=p^2+w; y=sqrt(abs(q)); x=x+t;

% This determines what type of eigenvalues have been determined
% ==================================================================
if q>0
    % A Real Pair
    if p<0
        y=-y;
        y=p+y; lambda(na)=x+y;lambda(n)=x-w/y;
    end
else
    % A Complex Pair
```

```
    lambda(na)=x+p+1i*y;lambda(n)=x+p-1i*y;
end
n=n-2; % This removes the last two rows and columns

% After a successful calculation, the procedure is restarted
[n H lambda p q r s t w x y z L m na its]...
    =nextw(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);

end
end

%% Continue 2:  Apply the QR Similarity Transformations
% ==========================================================================

% This subfunction does the bulk of the work.  It applies successive
% transformations to the original matrix to maintain upper Hessenberg form
% while forcing the appearance of the eigenvalues.

function [n H lambda p q r s t w x y z L na its]...
    =cont2(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its)

if n≠0
% This zeros the subdiagonal elements
% ==========================================================================
for i=m+2:1:n
    H(i,i-2)=0;
end
for i=m+3:1:n
    H(i,i-3)=0;
end

% Double QR step involving rows 'L' to 'n' and columns 'm' to 'n'
% ==========================================================================
% This applies what is called the 'double QR step'.  In doing so, all the
% calculations can be done with real numbers while the complex parts appear
% as well.
for k=m:1:na
    if k≠m
        p=H(k,k-1); q=H(k+1,k-1);
        if k≠na
```

```
            r=H(k+2,k-1);
        else
            r=0;
        end
        x=abs(p)+abs(q)+abs(r);

% Determining numeric zero
% ==========================================================================
% make sure the next step doesn't give problems.  basically it says
% if x==0 then restart the process, but with numeric roundoff there could
% be a problem with explicitly calling a trancendental number 'zero'
        if abs(x)≤eps
            [n H lambda p q r s t w x y z L m na its]...
                =nextit(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);
        end
        p=p/x; q=q/x; r=r/x;
    end
    s=sqrt(p^2+q^2+r^2);
    if p<0
        s=-s;
    end
    if k≠m
        H(k,k-1)=-s*x;
    elseif L≠m
        H(k,k-1)=-H(k,k-1);
    end
    p=p+s; x=p/s; y=q/s; z=r/s; q=q/p; r=r/p;

% Row Modification
% ==========================================================================
    for j=k:1:n
        p=H(k,j)+q*H(k+1,j);
        if k≠na
            p=p+r*H(k+2,j);
            H(k+2,j)=H(k+2,j)-p*z;
        end
        H(k+1,j)=H(k+1,j)-p*y;
        H(k,j)=H(k,j)-p*x;
    end
    if k+3<n
```

```
        j=k+3;
    else
        j=n;
    end

% Column Modification
% ==================================================================
    for i=L:1:j
        p=x*H(i,k)+y*H(i,k+1);
        if k≠na
            p=p+z*H(i,k+2);
            H(i,k+2)=H(i,k+2)-p*r;
        end
        H(i,k+1)=H(i,k+1)-p*q;
        H(i,k)=H(i,k)-p;
    end
end

% Continue with the next iteration
[n H lambda p q r s t w x y z L m na its]...
    =nextit(n,H,lambda,p,q,r,s,t,w,x,y,z,L,m,na,its);

end
end
```

B.25 The LZ Method

This algorithm calculates the spectrum of the real/complex upper Hessenberg matrix A_{ij} using LZ elimination transformations.

Algorithm B.25.1. *The LZ Method*

input : Matrix A_{ij}

output : The spectrum

```
%% LR Algorithm for Complex Matrices - 2
```

```
% ==================================================================

% This code has the complete algorithm given by Wilkinson.  It uses some
% built in Matlab commands, but in many cases the original code is given
% and commented out.  The eigenvectors don't calculate correctly though!!

% This code calculates the eigenvalues of a real or complex upper Hessenberg Matrix.
% The original matrix must be converted to upper hessenberg form using
% This form will likely return complex eigenvalues.  This code uses complex algebra
% to handle the matrix transformations.  If the original matrix
% is not Upper Hessenberg, then the matrix must first be conditioned by
% Complex_Nonsymmetric_to_Hessenberg.m

% This code is iterative, but without a general loop.  The iteration is
% performed when conditional statements point the information to different
% subfunctions.  Ultimately, the code will finish and break the loop.

% This algorithm follows Wilkinson's text 'Handbook for Automatic
% Computation Vol 2 Linear Algebra

% To automate this code in conjunction with simply
% input the upper Hessenberg matrix from 'A' as the
% input matrix 'A' in this program.  Matrix 'A' is preserved in the
% program.

% !if there is a problem passing variables remember that not all the
% functions are passing the same variables.

%% Initialize the Program
% ==================================================================
function []=Complex_LR_2()

clear;close all;clc;
tic;

% Define the Matrix
% ==================================================================
J=5;
A=rand(J)+1i*rand(J);
H=A; % Define 'H' as 'A' to preserve 'A'.  All calculations are done on 'H'
```

```
% Initialize Parameters
% ==================================================================
[m n]=size(H); % This automatically calculates the size of H
w=zeros(1,n); % This initializes the eigenvalue vector
its=0; % This is the iteration counter to determine the shifts or failure
t=0; % This is the 'exceptional shift' variable

% These are arbitrary placeholders for variables used later in the solution
s=0;x=0;y=0;z=0;nrm=0;M=0;

done=0;
% This cleanly stops the calculations once all the eigenvalues are found


% These are the integers produced by 'balance' if Wilkinson's algorithim is
% used.  low=1, upp=n if balancing has not been used or a different
% algorithm was employed.
low=1;upp=n;

% 'interchange' is an n X 1 array produced by complex hessenberg
% transformations.  If the primary matrix is already Hessenberg then set
% interchange(i)=i;

interchange=zeros(1,n);
for I=1:1:n
    interchange(I)=I;
end


% The following is from the algorithm.  It defines vr=eye(n) vi=zeros(n,n)
% for I=1:1:n
%     for J=1:1:n
%         vr(I,J)=0;
%         vi(I,J)=0;
%     end
%     vr(I,I)=1;
% end
% vr=eye(n);vi=zeros(n,n);
v=eye(n);
```

```
% Eigenvector Calculations
% ==========================================================================
for I=upp-1:-1:low+1
    J=interchange(I);
    for K=I+1:1:upp
        v(K,I)=H(K,I-1);
    end
    if I≠J
        for K=I:1:upp
            v(I,K)=v(J,K);
            v(J,K)=0;
        end
        v(J,I)=1+1i*imag(v(J,I));
    end
end
% ==========================================================================

% Isolated Roots
% ==========================================================================
% This section doesn't matter if I don't have 'low' and 'upp' defined from
% the complex hessenberg reduction routine.  I have no idea what the syntax
% means and can't really test it because i'm using a different balance
% algorithm - one that doesn't produce 'low' and 'upp'. It might be BOTH
% with the same commands.  It doesn't matter because without actual input
% for low and upp then those loops are never entered.

% This syntax is for ALGOL 60. I don't know what it actually means

for I=1:1:low-1
    w=Hr(I,I);
end

for I=upp+1:1:n;
    w=Hr(I,I);
end
% ==========================================================================
en=upp;

nextw(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);
end
```

```
%% Continuation Subfunction
% ==========================================================================

% This subfunction tests if all the eigenvalues have been solved.  If so it
% breaks the loop and displays the results.  If not it restarts the
% calculation to determine the next eigenvalue.

function [n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    nextw(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done)

if done==0

% This determines if the all the eigenvalues are calculated and ends the
% computation if they are.
% ==========================================================================
if en<low
    [n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    finish(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);
end
its=0;

% Begin the next iteration
% ==========================================================================
[n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    nextit(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);

end

end

%% Begin the next iteration
% ==========================================================================
function [n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    nextit(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done)

if done==0

% Look for a single small subdiagonal element
% ==========================================================================
```

```
% If a single small subdiagonal element is found, then the matrix is
% reduced to calculate the eigenvalues of the submatrix below the small
% subdiagonal.  This accelerates the convergence of the algorithm

for K=en:-1:low+1
    if abs(H(K,K-1))≤eps*(abs(H(K-1,K-1))+abs(H(K,K)))
        [n low upp interchange H w v I J M its en s t x,y z nrm done]=...
            cont1(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);
    end
end

K=low;
[n low upp interchange H w v I J M its en s t x,y z nrm done]=...
    cont1(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);

end

end

%% Continue 1:  Testing for Convergence
% ==================================================================

% This subfunction begins by testing if the eigenvalues are explicitly
% calculated because either the original matrix or a submatrix has be
% completely reduced and the eigenvalue appears explicitly on the diagonal
% elements.  If an eigenvalue is not determined, then the either an
% exceptional shift is performed to avoid redundant shifting or two
% consecutive subdiagonal elements are tested for.  If two are found, they
% constitute the same affect as a single small subdiagonal.  This
% subfunction continues until the possible acceleration techniques are
% exhausted.  At this point, it continues on as normal.

function [n low upp interchange H w v I J M its en s t x,y z nrm done]=...
    cont1(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done)

if done==0;

% This determines if a root has been found
% ==================================================================
if K==en
```

```
    [n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
        root(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);
end % maybe could use elseif

% This determines if the calculation has failed
% ==================================================================
if its==30
    disp('Max iterations exceeded. Calculations failed.')
    done=1;
end

% Form Exceptional Shift
% ==================================================================
if its==30
    disp('Convergence Failed')
    done=1;
elseif its==10 || its==20
    s=abs(H(en,en-1))+abs(H(en-1,en-2));
else
    s=H(en,en);
    x=H(en-1,en)*H(en,en-1);
    if abs(x)≠0
        y=(H(en-1,en-1)-s)/2;
        z=sqrt(y);

% The following calculates the same value of 'z' as the code above without
% using any matlab internal commands
% ==================================================================
%           z=csqrt(real(y),imag(y));
% ==================================================================

        if real(y)*real(z)+imag(y)*imag(z)<0
            z=-z;
        end

        x=x/(y+z);

% The following calculates the same value of 'x' as the code above without
% using any matlab internal commands.  Be sure to comment out the above
% command before trying to use the following code
```

```
% ==========================================================================
%           x=cdiv(real(x),imag(x),real(y)+real(z),imag(y)+imag(z));
% ==========================================================================
        s=s-x;
    end
end

for I=low:1:en
    H(I,I)=H(I,I)-s; % This modifies some of the diagonal elements only
end
t=t+s; % This is the exceptional shift
its=its+1; J=K+1; % this increments the iteration counter

% Look for two consecutive small subdiagonal elements
% ==========================================================================
% If two consecutive small subdiagonal element are found, then the matrix is
% reduced to calculate the eigenvalues of the submatrix below the small
% subdiagonal elements.  This accelerates the convergence of the algorithm

% These are real numbers
x=abs(H(en-1,en-1));
y=abs(H(en,en-1));
z=abs(H(en,en));
for M=en-1:-1:J
    y=real(y)+1i*real(y);
    y=abs(H(M,M-1))+1i*imag(y);
    x=real(x)+1i*real(z);
    z=real(x)+1i*imag(z);
    x=abs(H(M-1,M-1))+1i*imag(x);
    if real(y)≤eps*real(z)/imag(y)*(real(z)+real(x)+imag(x))
        [n low upp interchange H w v I J K its en s t x,y z nrm done]=...
        cont2(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);
    end
end
M=K;

[n low upp interchange H w v I J K its en s t x,y z nrm done]=...
    cont2(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);

end
```

```
end

%% Continuation 2:  Apply the Transformations
% ==========================================================================

% This subfunction does the bulk of the work.  It applies successive
% transformations to the original matrix to maintain upper Hessenberg form
% while forcing the appearance of the eigenvalues.

function [n low upp interchange H w v I J K its en s t x,y z nrm done]=...
    cont2(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done)

if done==0

% Triangular Decomposition H=LxR
% ==========================================================================
for I=M+1:1:en
    x=H(I-1,I-1);
    y=H(I,I-1);

%     I think I can write modulus for the next conditional
% ==========================================================================
%     if abs(real(x))+abs(imag(x))<abs(real(y))+abs(imag(y))
    if abs(x)<abs(y)

% Interchange rows of H
% ==========================================================================
        for J=I-1:1:n
            z=H(I-1,J);H(I-1,J)=H(I,J);H(I,J)=z;
        end
% This is the vectorized version of the above for loop. It is slower though
% ==========================================================================
%         z=H(I-1,I-1:n);H(I-1,I-1:n)=H(I,I-1:n);H(I,I-1:n)=z;
% ==========================================================================
        z=x/y; w(I)=1;

% The following calculates the same value of 'z' as the code above without
% using any matlab internal commands
% ==========================================================================
```

```
%           z=cdiv(real(x),imag(x),real(y),imag(y)); w(I)=1;
% ==================================================================

    else
        z=y/x; w(I)=-1;

% The following calculates the same value of 'z' as the code above without
% using any matlab internal commands
% ==================================================================
%           z=cdiv(real(y),imag(y),real(x),imag(x)); w(I)=-1;
% ==================================================================

    end

    H(I,I-1)=z;
    for J=I:1:n
        H(I,J)=H(I,J)-z*H(I-1,J);
    end

% The following is how the text is written does the same calculation as the
% previous loop
% ==================================================================
%       for J=I:1:n
%           hr=real(H(I,J))-real(z)*real(H(I-1,J))+imag(z)*imag(H(I-1,J));
%           hi=imag(H(I,J))-real(z)*imag(H(I-1,J))-imag(z)*real(H(I-1,J));
%           H(I,J)=hr+i*hi;
%       end
% ==================================================================
end

% Composition RxL=H
% ==================================================================
for J=M+1:1:en
    x=H(J,J-1); H(J,J-1)=0;

% Interchange Columns of H and v if necessary
% ==================================================================
    if real(w(J))>0
        for I=1:1:J
            z=H(I,J-1); H(I,J-1)=H(I,J); H(I,J)=z;
```

```
        end
        for I=low:1:upp
            z=v(I,J-1); v(I,J-1)=v(I,J); v(I,J)=z;
        end
    end
% ==================================================================

% Begin Accumulated Transformation
% ==================================================================

    for I=1:1:J
        H(I,J-1)=H(I,J-1)+x*H(I,J);
    end

% The following is how the text is written does the same calculation as the
% previous loop
% ==================================================================
%     for I=1:1:J
%         hr=real(H(I,J-1))+real(x)*real(H(I,J))-imag(x)*imag(H(I,J));
%         hi=imag(H(I,J-1))+real(x)*imag(H(I,J))+imag(x)*real(H(I,J));
%         H(I,J-1)=hr+i*hi;
%     end
% ==================================================================

    for I=low:1:upp
        v(I,J-1)=v(I,J-1)+x*v(I,J);
    end
% End Accumulated Transformation
% ==================================================================
end

% Begin next iteration
[n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    nextit(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);

end

end

%% Eigenvalues are determined
```

```
% ==========================================================================

% This subfunction simply stores the eigenvalues

function[n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    root(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done)

if done==0

% A root found
% ==========================================================================
w(en)=H(en,en)+t; w(en)=real(w(en))-1i*imag(w(en));
en=en-1;

% Begin looking for the next eigenvalue
[n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    nextw(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done);

end

end

%% Finish
% ==========================================================================
% The 'finish' function is called once all the eigenvalues have been
% accounted for.  It also back calculates the eigenvectors of the original
% matrix from backsubstitution and transformations

function[n low upp interchange H w v I J K M its en s t x,y z nrm done]=...
    finish(n,low,upp,interchange,H,w,v,I,J,K,M,its,en,s,t,x,y,z,nrm,done)

if done==0

% All roots found
% ==========================================================================
nrm=0;
for I=1:1:n
    nrm=nrm+abs(real(w(I)))+abs(imag(w(I)));
    for J=I+1:1:n
        nrm=nrm+abs(real(H(I,J)))+abs(imag(H(I,J)));
```

```
    end
end

% Backsubstitute to find vectors of upper triagular form
% ==================================================================

for en=n:-1:2
    x=w(en);
    for I=en-1:-1:1
        z=H(I,en);
        for J=I+1:1:en-1
            z=z+H(I,J)*H(J,en);
        end
        y=x-w(I);
        if abs(y)==0
            y=eps*nrm+1i*imag(y);
        end
        H(I,en)=z/y;

% The following calculates the same value of 'H(I,en)' as the code
% above without using any matlab internal commands
% ==================================================================
%           H(I,en)=cdiv(real(z),imag(z),real(y),imag(y));
% ==================================================================

    end
end
% End Backsubstitute
% ==================================================================

% Multiply by transformation matrix to give vectors of original full matrix
% ==================================================================
% This section doesn't matter if I don't have 'low' and 'upp' defined from
% the complex hessenberg reduction routine.  I have no idea what the syntax
% means and can't really test it because i'm using a different balance
% algorithm - one that doesn't produce 'low' and 'upp'.  It should never be
% called without the original 'balance' routine.  It might be BOTH with the
% same command

for I=1:1:low-1
```

```
    for J=I+1:1:n
        v(I,J)=H(I,J);
    end
end

for I=upp+1:1:n;
    for J=I+1:1:n
        v(I,J)=H(I,J);
    end
end % vectors of isolated roots

for J=n:-1:low
    for I=low:1:upp
        z=v(I,J);
        if upp<J
            M=upp;
        else
            M=J-1;
        end
        for K=low:1:M
            z=z+v(I,K)*H(K,J); % not as coded, using MatLab syntax
        end
        v(I,J)=z;
    end
end

disp('Calculation Successful')
format long % To increase the displayed digits
% Display the Eigenvalues
% ==========================================================================
disp('The Eigenvalues are')
disp(w')

% Display the Eigenvectors
% ==========================================================================
% disp('The Eigenvectors are')
% disp(v')
its_test=0;
if its==10 || its==20
    its_test=1;
```

```
end
if en≤2 && its_test==1
    disp('Convergence occured in a nontypical way.')
    disp('It may have occured on an iteration flagged to apply a shift')
    disp('If this is the case, the calculation should be correct,')
    disp('but use caution')
end
format short
done=1;
toc

end

end


%% Appendix
% ==========================================================================
% The following subfunctions are explicitly programed routines to calculate
% the complex square root and complex division.  Matlab has these built in
% and I am using them now.  These were just to test and make sure I hadn't
% forgotten my complex algebra and that I understood the ALGOL 60 syntax.

function z=csqrt(xr,xi)

% This section of code is the given algorithm for csqrt from the text.  It
% matches with my shortened version above.  I'm not 100% sure i'm
% calculating the correct values though.  The inputs are confusing.
% ==========================================================================

h=sqrt((abs(xr)+abs(xr+1i*xi))/2);
if xi≠0
    xi=xi/(2*h);
end

if xr≥0
    xr=h;
else
    if xi≥0
        xr=xi;xi=h;
```

```
    else
        xr=-xi;xi=-h;
    end
end
z=xr+1i*xi;

% ==================================================================

end

function z=cdiv(xr,xi,yr,yi)

% Be careful with the inputs.  This code is general and uses some of the
% same variables as the main code.  They don't necessarily mean anything.
% Just trust the code is correct.  The basic order to call will be
% z=cdiv(y,x) is the same as z=y/x
% ==================================================================

if abs(yr)>abs(yi)
    h=yi/yr;yr=h*yi+yr;zr=(xr+h*xi)/yr;zi=(xi-h*xr)/yr;
else
    h=yr/yi;yi=h*yr+yi;zr=(h*xr+xi)/yi;zi=(h*xi-xr)/yi;
end
z=zr+1i*zi;
% ==================================================================

end
```

B.26 The Generalized LZ Algorithm

This algorithm calculates the spectrum of the real/complex matrix pencil $A_{ij} - \lambda B_{ij}$ for the generalized eigenvalue problem.

Algorithm B.26.1. *The Generalized LZ Algorithm*

input : The upper Hessenberg matrix A_{ij} and the upper triangular matrix B_{ij}

output : The spectrum

```
%% The LZ Algorithm for Solving Eigenvalues of Ax=lambda*Bx REDUX
% ==========================================================================
% This code is designed to find the eigensystem for two complex matrices A
% and B.  It is translated from Kaufman-The LZ Algorithm to Solve the
% Generalized Eigenvalue Problem for Complex Matrices.  It consists of two
% separate subroutines: LZ_HESS and LZ_IT.  LZ_HESS is called to reduce A to
% upper Hessenberg and B to triangular form outputting the reduced
% eigensystem.  LZ_IT  returns the diagonal elements of the trangularized
% matrices.  The eigenvalues of the 'system' are found by dividing the
% corresponding diagonal elements eig(A)/eig(B).  The eigenvectors produced
% by LZ_IT are normalized so the modulus of the largest component is 1.

% The eigenvectors of the problem are computed by computing the
% eigenvectors of the triangularized A and B matrices and multiplying them
% by the product of all column transformations.

% This solver works and gives accuracy compared to the built in matlab
% solver of 10^(-11) as the absolute worst case.  Typically 10^(-14)

% Since the shift is calculated as the eigenvalue approximation of the
% lower 2x2 submatrix, the first eigenvalue to converge should be at the
% bottom.  This algorithm deflates from the bottom up.  It also
% incorporates a new strategy for block decomposition

% This version has a different generalized balance algorithm.  It handles
% infinity and NaN by replacing them by a number after the D matrices are
% determined
%% Initialize the Program
% ==========================================================================

% Define the Matrices, initialize the variables, and test to ensure that
% size(A)=size(B)
function []=Generalized_LZ(A,B)

 clear;close all;clc;


% Define the Matrices
% ==========================================================================
% -------------------------------------------------------------------------
```

```
% d=.25;
% J=10;
% A=rand(J)+1i*rand(J);
% B=rand(J)+1i*rand(J);
% A=full(sprand(J,J,d)+1i*sprand(J,J,d));
% B=full(sprand(J,J,d)+1i*sprand(J,J,d));

% A=magic(4);B=pascal(4)^(-1);
A=rand(1000);B=rand(1000);
tic;
ORIGINAL=eig(A,B);
disp('The original eigenvalues are')
disp(ORIGINAL);
toc
tic;
% Initialize Parameters
% ==================================================================
N=size(A,1); % N is the size of A. It is deflated as the solution converges
SIZE=N;
EIGEN=zeros(N,1);
[A B N]=LZ_HESS(A,B,N);
disp(norm(A,'fro'));
disp(norm(B,'fro'))
[A B]=Generalized_Balance(A,B);
[A B]=B_Mod(A,B,N);

ITS=0;  % This is to initialize the iteration counter before the reductions
SHIFT=0;
M=1;
while N>0
    [A B N M ITS SHIFT EIGEN]...
        =LZ_CONVERGENCE(A,B,N,M,ITS,SHIFT,EIGEN);
    if N==1
        break;
    end
    [A B N M ITS SHIFT]...
        =LZ_SHIFT(A,B,N,M,ITS,SHIFT,EIGEN,SIZE);
    [A B N M]...
        =LZ_TRANSFORMS(A,B,N,M);
%       [A B]=Generalized_Balance(A,B);
```

```
end
    EIGEN(1)=A(1,1)/B(1,1)+SHIFT;
for K=1:SIZE
    if abs(EIGEN(K))≥10^7
        EIGEN(K)=Inf;
    elseif abs(EIGEN(K))≤10^(-6)
        EIGEN(K)=0;
    elseif isnan(abs(EIGEN(K)))==1
        EIGEN(K)=NaN;
    end
end

disp('The Eigenvalues are')
disp(EIGEN)
disp(max(sort(abs(ORIGINAL))-sort(abs(EIGEN))))
disp('done')
toc
end

%% Convergence Test
% ==================================================================
% This section will test for convergence at the current bottom right corner as
% well as test for possible double root or block decomposition possibilities to
% increase convergence speed.
function [A B N M ITS SHIFT EIGEN]=LZ_CONVERGENCE(A,B,N,M,ITS,SHIFT,EIGEN)

FLAG=0;
while FLAG==0;
    FLAG=1;
    for K=N:-1:2
        if abs(A(K,K-1))≤10^(-15)
            break;
        end
    end
     % This is so iterations are only added when decompositions don't occur
    if K==M+1
        ITS=ITS+1;
    else
        M=K-1;
        ITS=0;
```

```
    end

    if M==N-1
        if abs(B(N,N))≤eps
            B(N,N)=0;
        end
        EIGEN(N)=A(N,N)/B(N,N)+SHIFT;          % A single root has been found
        N=N-1;                                 % Deflate one row and one column
        M=1;                                                          % Reset M
        ITS=0;                                 % The counter is reinitialized
        if N>2;
            FLAG=0;
        end
    elseif M==N-2 && N≠3 && A(N,N-1)...
            ≤sqrt(eps)            % SPECIAL CASE: A 2x2 block decomposition
        detB2x2=B(N-1,N-1)*B(N,N)-B(N-1,N)*B(N,N-1);
        if abs(detB2x2)>eps;
            BI=1/(detB2x2)*...
                [B(N,N) -B(N-1,N);-B(N,N-1) B(N-1,N-1)];
                        % BI is the inverse of the lower 2x2 submatrix of B
            C=BI*A(N-1:N,N-1:N);               % This is A_{2,2}*B_{2,2}^{-1}
            Z=sqrt((C(1,1)+C(2,2))^2-4*(C(1,1)*C(2,2)-C(1,2)*C(2,1)));
                    % Z is the square root of the characteristic polynmial
            EIGEN(N)=.5*(C(1,1)+C(2,2)+Z)+SHIFT;
            EIGEN(N-1)=.5*(C(1,1)+C(2,2)-Z)+SHIFT;
                                               % A double root has been found
            N=N-2;                             % Deflate 2 rows and 2 columns
            M=1;                                                          % Reset M
            ITS=0;                             % The counter is reinitialized
            if N>2
                FLAG=0;
            end
        end
    end

end
end

%% Shift
% ==================================================================
```

```
% This function forms the shift required to accelerate convergence and to
% allow for imaginary eigenvalues to be determined

function [A B N M ITS SHIFT]=LZ_SHIFT(A,B,N,M,ITS,SHIFT,EIGEN,SIZE)
MAX=1000;
if ITS==MAX
    disp('Max iterations exceeded.  Calculations failed.')
    disp('The calculation is expected to be inaccurate.')
    for K=1:SIZE
        if abs(EIGEN(K))≥10^8
            EIGEN(K)=Inf;
        elseif abs(EIGEN(K))≤10^(-6)
            EIGEN(K)=0;
        end
    end
    disp('The number of successfully computed Eigenvalues is')
    disp(SIZE-N);
    disp('The Successfully Computed Eigenvalues are')
    disp(EIGEN(N:SIZE))
    N=1;
    S=0;
elseif ITS==1*MAX || ITS==2*MAX || ITS==3*MAX || ITS==4*MAX
%     Form ad-hoc Shift
% ---------------------------------------------------------------------
    S=abs(A(N,N-1))+abs(A(N-1,N-2));                    % Bottom right corner
else
% ---------------------------------------------------------------------
%     Form Exceptional Shift
%     The shift is calculated as the approximation to the eigenvalue of the
%     lower 2x2 submatrix of A*B^(-1)
%     Force the shift to converge to the eigenvalue in the bottom corner
%     so that the eigenvalue is always deflated from the bottom right up.
% ---------------------------------------------------------------------
    detB2x2=B(N-1,N-1)*B(N,N)-B(N-1,N)*B(N,N-1);
    if abs(detB2x2)>eps
        BI=1/(detB2x2)*...
            [B(N,N) -B(N-1,N);-B(N,N-1) B(N-1,N-1)];
            % BI is the inverse of the lower 2x2 submatrix of B
        C=BI*A(N-1:N,N-1:N); % This is A_{2,2}*B_{2,2}^{-1}
        Z=sqrt((C(1,1)+C(2,2))^2-4*(C(1,1)*C(2,2)-C(1,2)*C(2,1)));
```

```
            % Z is the square root of the characteristic polynmial
        S1=.5*(C(1,1)+C(2,2)+Z);
        S2=.5*(C(1,1)+C(2,2)-Z);
              % Below develop the conditionals to ensure convergence to the
                                                  % bottom right corner only.
        D(1)=abs(C(2,2)-S1);
        D(2)=abs(C(2,2)-S2);
        [D I]=min(D);
        if I==1
            S=S1;
        else
            S=S2;
        end
    elseif N>2
        S=abs(A(N,N-1))+abs(A(N-1,N-2));                % Bottom right corner
    else
        S=0;
    end
end
% ---------------------------------------------------------------------
%     Apply the shift
% ---------------------------------------------------------------------
if isnan(S)≠1
    SHIFT=SHIFT+S;
    A=A-S*B;
end

end

%% Transforms
% ===================================================================
% This function applies the row and column transforms to A and B

function [A B N M]=LZ_TRANSFORMS(A,B,N,M)

NZERO=eps^(1);                                      % NZERO is numeric zero.
for K=M:1:N-1
% ---------------------------------------------------------------------
% Annihilate the subdiagonal element in A with row transformations
% Apply the transformation to both A and B
```

```
% ----------------------------------------------------------------------
    if A(K,K)==0
        A(K,K)=NZERO; DIV=A(K,K);
    else
        DIV=A(K,K);
    end
        Y=A(K+1,K)/DIV;
        A(K+1,K:N)=A(K+1,K:N)-Y*A(K,K:N);
        B(K+1,K:N)=B(K+1,K:N)-Y*B(K,K:N);
% ----------------------------------------------------------------------
% Restore B to triangular form with column transformations
% Apply the transformation to both B and A
% ----------------------------------------------------------------------
    if B(K+1,K+1)==0
        B(K+1,K+1)=-NZERO; DIV=B(K+1,K+1);
    else
        DIV=B(K+1,K+1);
    end
        Y=B(K+1,K)/DIV;
        A(M:K+1,K)=A(M:K+1,K)-Y*A(M:K+1,K+1);
        B(M:K+1,K)=B(M:K+1,K)-Y*B(M:K+1,K+1);
end

end
```

Appendix C

Unstable Frequencies

The spectral results for the biglobal analysis are densely populated, making the identification of unstable circular frequencies difficult. While most spectral figures in Ch. 6 show eigenvalues above the critical line, many of these eigenvalues appear at similar circular frequencies as damped eigenvalues. The overall stability of a discrete circular frequency is contingent on the contribution of all eigenvalues at or near its value. As a means of data reduction and post processing, the spectral results of the parametric studies are analyzed to identify small intervals where the overall growth is expected to be positive. This post processing technique is similar to computing a histogram where eigenvalues falling within a predefined small interval are collected. Rather than seeking the distribution of circular frequencies, the growth rates of the collected eigenvalues are averaged to obtain a distributed growth rate, $\tilde{\omega}_i$. In this way, more quantitative trends can be identified within the spectrum. These results are gathered in the subsequent sections and can be referenced to their corresponding figures in Ch. 6. Frequencies are tested at intervals of 2.5 ± 2.5 for all tables. The general equation used to calculate the distributed growth rate is given as:

$$\tilde{\omega}_i = \frac{1}{N} \sum_{j=1}^{N} (\omega_i)_j \tag{C.1}$$

where N is the number of eigenvalues falling in the interval $a \leq \omega_r \leq b$.

C.1 The Complex-Lamellar Bidirectional Vortex

This section tabulates the predicted unstable circular frequencies for the complex-lamellar bidirectional vortex.

C.1.1 Axisymmetric Spectrum

Table C.1: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the axisymmetric complex-lamellar parametric study in Fig. 6.3 for $\kappa = 0.1$.

$\kappa =$	0.1		0.01	0.001
	132.5	2.5	None	None
	157.5	7.2		
	177.5	20.9		
	195.0	17.0		
	225.0	18.6		
	252.5	5.9		
	255.0	44.3		
	257.5	41.9		
	260.0	21.0		
	262.5	15.2		

$Re =$	10000		5000	1000
	132.5	2.5	None	None
	157.5	7.2		
	177.5	20.9		
	195.0	17.0		
	225.0	18.6		
	252.5	5.9		
	255.0	44.3		
	257.5	41.9		
	260.0	21.0		
	262.5	15.2		

$l =$	2.5		1.5		0.5
	130.0	0.2	125.0	1.2	None
	155.0	14.5	127.5	5.3	
	177.5	3.5	130.0	1.8	
	180.0	7.4	162.5	4.8	
	195.0	42.2	180.0	23.4	
	220.0	10.1	250.0	41.9	
	255.0	35.8	257.5	35.1	
	257.5	45.1	260.0	25.9	
	260.0	48.1	262.5	4.6	
	262.5	59.2	265.0	27.3	

$m =$	0		2		4	
	132.5	2.5	97.5	4.8	82.5	1.9
	157.5	7.2	117.5	5.5	85.0	0.2
	177.5	20.9	120.0	2.1	97.5	0.3
	195.0	17.0	140.0	1.0	112.5	0.5
	225.0	18.6	147.5	0.9	140.0	0.4
	252.5	5.9	157.5	3.3	142.5	0.1
	255.0	44.3	165.0	8.6	155.0	2.1
	257.5	41.9	167.5	5.3	157.5	3.3
	260.0	21.0	182.5	5.4	165.0	3.6
	262.5	15.2	185.0	0.5	167.5	4.2

C.1.2 Asymmetric Spectrum

Table C.2: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for variation in κ (Fig. 6.4).

$\kappa =$	0.1		0.01	0.001
	67.5	1.0	None	None
	77.5	2.2		
	95.0	2.1		
	97.5	3.4		
	130.0	4.5		
	132.5	2.6		
	135.0	1.1		
	137.5	2.9		
	140.0	0.3		
	157.5	6.6		

Table C.3: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for variation in Re (Fig. 6.5).

	$\kappa = 0.1$					$\kappa = 0.01$			
$Re =$	10000		5000	1000	$Re =$	10000		5000	1000
	67.5	1.0	None	None		None		None	None
	77.5	2.2							
	95.0	2.1							
	97.5	3.4							
	130.0	4.5							
	132.5	2.6							
	135.0	1.1							
	137.5	2.9							
	140.0	0.3							
	157.5	6.6							

Table C.4: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for variation in l (Fig. 6.9).

	$\kappa = 0.1$						$\kappa = 0.01$				
$l =$	2.5			1.5		0.5	$l =$	2.5		1.5	0.5
	65.0	0.2	85.0	0.0	117.5	0.5		90.0	0.1	None	None
	85.0	1.1	87.5	1.1	155.0	0.1		332.5	1.0		
	87.5	2.0	90.0	1.9	180.0	10.3					
	95.0	1.4	97.5	0.3	182.5	35.4					
	97.5	1.7	130.0	5.4	215.0	38.9					
	105.0	0.1	132.5	5.2	247.5	3.3					
	107.5	3.1	135.0	2.5	250.0	7.1					
	127.5	1.3	155.0	3.1	282.5	40.7					
	130.0	2.4	157.5	2.4	315.0	1.5					
	132.5	5.5	160.0	4.8	317.5	29.4					

Table C.5: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric complex-lamellar spectrum for multidirectional flow number, m (Fig. 6.11).

	$\kappa = 0.1$							$\kappa = 0.01$				
$m =$	0		2		4		$m =$	0	2		4	
	67.5	1.0	82.5	2.6	75.0	2.0		None	90.0	0.0	75.0	0.5
	77.5	2.2	92.5	1.1	87.5	0.8			105.0	1.9	92.5	0.1
	95.0	2.1	95.0	8.5	100.0	2.8			107.5	2.3	102.5	0.7
	97.5	3.4	107.5	1.6	102.5	8.9			110.0	0.6	127.5	2.1
	130.0	4.5	117.5	3.2	112.5	5.1			115.0	1.4	135.0	3.1
	132.5	2.6	120.0	0.5	142.5	6.4			117.5	3.5	137.5	7.0
	135.0	1.1	125.0	4.7	152.5	8.9			125.0	0.3	142.5	2.1
	137.5	2.9	127.5	8.4	155.0	18.0			127.5	0.3	150.0	6.8
	140.0	0.3	145.0	3.0	157.5	10.8			130.0	2.3	152.5	8.0
	157.5	6.6	147.5	1.9	160.0	12.5			132.5	1.0	155.0	2.3

C.2 The Linear Beltramian Bidirectional Vortex

This section tabulates the predicted unstable circular frequencies for the linear Beltramian bidirectional vortex.

C.2.1 Axisymmetric Spectrum

Table C.6: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the axisymmetric linear Beltramian parametric study in Fig. 6.14 for $\kappa = 0.1$.

$\kappa =$	0.1		0.01		0.001
	72.5	1.6	295.0	0.3	None
	75.0	2.1	455.0	5.1	
	107.5	7.2			
	110.0	3.9			
	112.5	5.2			
	125.0	7.9			
	135.0	15.7			
	137.5	5.2			
	142.5	9.3			
	145.0	15.9			

$Re =$	10000		5000	1000
	72.5	1.6	None	None
	75.0	2.1		
	107.5	7.2		
	110.0	3.9		
	112.5	5.2		
	125.0	7.9		
	135.0	15.7		
	137.5	5.2		
	142.5	9.3		
	145.0	15.9		

$l =$	2.5		1.5		0.5
	72.5	1.7	102.5	0.5	None
	75.0	0.8	105.0	7.6	
	112.5	2.3	117.5	0.8	
	115.0	16.4	120.0	22.1	
	117.5	0.5	122.5	26.0	
	132.5	0.4	135.0	8.8	
	142.5	9.5	137.5	7.8	
	152.5	5.7	145.0	6.8	
	155.0	18.5	147.5	0.5	
	207.5	17.3	152.5	9.1	

$m =$	0		2		4	
	72.5	1.6	22.5	0.1	40.0	2.5
	75.0	2.1	40.0	0.9	42.5	3.2
	107.5	7.2	50.0	1.1	70.0	9.1
	110.0	3.9	77.5	2.2	90.0	7.3
	112.5	5.2	92.5	2.5	92.5	1.1
	125.0	7.9	95.0	5.5	100.0	3.4
	135.0	15.7	105.0	5.1	102.5	8.2
	137.5	5.2	107.5	11.3	125.0	2.6
	142.5	9.3	110.0	1.9	127.5	4.4
	145.0	15.9	117.5	3.4	135.0	16.1

C.2.2 Asymmetric Spectrum

Table C.7: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for variation in κ (Fig. 6.15).

$\kappa =$	0.1		0.01		0.001
	72.5	0.6	92.5	0.3	None
	105.0	1.3	100.0	1.8	
	110.0	0.6	102.5	0.2	
	112.5	4.2	107.5	0.1	
	125.0	5.1	152.5	0.2	
	127.5	5.4	175.0	0.3	
	130.0	1.3	182.5	1.0	
	135.0	3.8	187.5	0.9	
	137.5	14.3	190.0	0.8	
	140.0	1.7	535.0	0.2	

Table C.8: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for variation in Re (Fig. 6.16).

$\kappa = 0.1$					$\kappa = 0.01$				
$Re =$	10000		5000	1000	$Re =$	10000		5000	1000
	72.5	0.6	None	None		92.5	0.3	None	None
	105.0	1.3				100.0	1.9		
	110.0	0.6				102.5	0.2		
	112.5	4.2				107.5	0.1		
	125.0	5.1				152.5	0.2		
	127.5	5.4				175.0	0.3		
	130.0	1.3				182.5	1.0		
	135.0	3.8				187.5	0.9		
	137.5	14.3				190.0	0.8		
	140.0	1.7				535.0	0.3		

Table C.9: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for variation in l (Fig. 6.20).

$\kappa = 0.1$							$\kappa = 0.01$					
$l =$	2.5		1.5		0.5		$l =$	2.5		1.5		0.5	
	72.5	0.8	57.5	0.7	182.5	0.1		87.5	0.0	107.5	1.3	77.5	0.7
	75.0	1.6	97.5	1.0	227.5	0.3		90.0	0.0			87.5	2.2
	100.0	0.0	110.0	3.4	262.5	0.5		100.0	0.2			90.0	6.5
	110.0	3.4	122.5	2.3	265.0	2.8		130.0	0.5			117.5	5.0
	112.5	0.4	125.0	2.5	270.0	0.7		177.5	0.7			125.0	1.8
	115.0	1.7	130.0	0.9	282.5	19.6		180.0	0.8			127.5	6.9
	117.5	0.8	137.5	4.4	285.0	36.9		182.5	1.4			145.0	7.7
	127.5	2.4	160.0	0.7	302.5	25.5		190.0	6.5			152.5	6.1
	130.0	4.8	197.5	17.7	317.5	23.7		497.5	0.6			185.0	3.5
	132.5	3.1	200.0	22.9	320.0	75.4		540.0	2.4			192.5	1.0

Table C.10: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric linear Beltramian spectrum for multidirectional flow number, m (Fig. 6.21).

	$\kappa = 0.1$							$\kappa = 0.01$					
$m =$	0		2		4		$m =$	0		2		4	
	72.5	0.6	65.0	0.1	52.5	2.0		92.5	0.3	87.5	1.7	62.5	1.1
	105.0	1.3	115.0	9.7	67.5	5.5		100.0	1.9	90.0	0.2	77.5	2.9
	110.0	0.6	135.0	0.9	70.0	7.1		102.5	0.2	100.0	1.0	92.5	0.5
	112.5	4.2	137.5	11.2	72.5	2.5		107.5	0.1	107.5	2.5	122.5	1.1
	125.0	5.1	155.0	6.8	95.0	7.9		152.5	0.2	110.0	1.0	125.0	4.8
	127.5	5.4	157.5	21.3	110.0	0.2		175.0	0.3	112.5	0.0	127.5	11.8
	130.0	1.3	167.5	6.3	112.5	0.0		182.5	1.0	120.0	3.3	140.0	2.5
	135.0	3.8	170.0	4.1	120.0	2.4		187.5	0.9	137.5	0.7	165.0	0.3
	137.5	14.3	185.0	27.4	122.5	14.8		190.0	0.8	147.5	5.6	182.5	1.3
	140.0	1.7	187.5	20.2	125.0	18.3		535.0	0.3	150.0	0.1	197.5	14.7

C.3 The Harmonic Beltramian Bidirectional Vortex

This section tabulates the predicted unstable circular frequencies for the harmonic Beltramian bidirectional vortex.

C.3.1 Axisymmetric Spectrum

Table C.11: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the axisymmetric harmonic Beltramian parametric study in Fig. 6.25 for $\kappa = 0.1$.

$\kappa =$	0.1		0.01		0.001
	72.5	1.0	295.0	0.3	None
	97.5	2.7	367.5	0.0	
	110.0	3.7	455.0	5.3	
	117.5	5.6			
	130.0	11.8			
	137.5	2.8			
	140.0	14.9			
	142.5	8.4			
	145.0	7.2			
	152.5	7.8			

$Re =$	10000		5000	1000	
	72.5	1.0	None	395.0	0.0
	97.5	2.7		397.5	0.0
	110.0	3.7		520.0	4.1
	117.5	5.6		522.5	4.1
	130.0	11.8		652.5	5.8
	137.5	2.8		655.0	5.8
	140.0	14.9			
	142.5	8.4			
	145.0	7.2			
	152.5	7.8			

$l =$	2.5		1.5		0.5
	70.0	1.2	80.0	2.2	None
	72.5	0.1	105.0	1.2	
	100.0	2.6	117.5	6.4	
	117.5	10.2	120.0	10.9	
	120.0	7.8	122.5	1.9	
	127.5	3.7	130.0	9.0	
	130.0	6.7	137.5	3.0	
	137.5	4.7	152.5	10.1	
	145.0	4.3	160.0	2.3	
	147.5	7.0	167.5	11.9	

$m =$	0		2		4	
	72.5	1.0	35.0	1.4	60.0	3.6
	97.5	2.7	37.5	3.9	62.5	7.9
	110.0	3.7	40.0	1.4	65.0	0.5
	117.5	5.6	90.0	11.5	90.0	2.2
	130.0	11.8	92.5	6.2	110.0	2.6
	137.5	2.8	122.5	9.5	120.0	1.2
	140.0	14.9	125.0	0.3	135.0	5.2
	142.5	8.4	130.0	3.2	137.5	1.3
	145.0	7.2	137.5	0.3	142.5	16.8
	152.5	7.8	150.0	8.4	145.0	7.7

C.3.2 Asymmetric Spectrum

Table C.12: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for variation in κ (Fig. 6.26).

$\kappa =$	0.1		0.01		0.001
	105.0	0.9	90.0	0.5	None
	112.5	3.4	92.5	1.6	
	115.0	3.1	100.0	1.7	
	120.0	5.1	102.5	0.6	
	127.5	5.2	135.0	0.1	
	130.0	0.2	152.5	0.1	
	135.0	13.7	182.5	0.9	
	137.5	9.4	187.5	0.8	
	145.0	8.2	190.0	0.8	
	147.5	3.1	535.0	0.3	

Table C.13: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for variation in Re (Fig. 6.27).

$\kappa = 0.1$					$\kappa = 0.01$				
$Re =$	10000		5000	1000	$Re =$	10000		5000	1000
	105.0	0.9	None	None		90.0	0.6	None	None
	112.5	3.4				92.5	1.7		
	115.0	3.1				100.0	1.7		
	120.0	5.1				102.5	0.6		
	127.5	5.2				135.0	0.2		
	130.0	0.2				152.5	0.2		
	135.0	13.7				182.5	1.0		
	137.5	9.4				187.5	0.9		
	145.0	8.2				190.0	0.8		
	147.5	3.1				535.0	0.4		

Table C.14: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for variation in l (Fig. 6.31).

$\kappa = 0.1$							$\kappa = 0.01$					
$l =$	2.5		1.5		0.5		$l =$	2.5		1.5		0.5	
	75.0	0.3	80.0	1.0	155.0	0.2		87.5	1.3	107.5	1.3	77.5	0.6
	77.5	0.1	110.0	0.2	157.5	1.5		100.0	0.3			87.5	2.1
	82.5	0.1	112.5	0.0	282.5	23.3		130.0	0.5			90.0	6.5
	102.5	0.4	122.5	0.6	285.0	66.2		162.5	0.5			117.5	0.6
	112.5	0.6	125.0	2.1	317.5	17.8		177.5	1.2			125.0	1.8
	117.5	3.9	127.5	0.2	320.0	23.0		180.0	0.8			127.5	6.9
	120.0	4.5	135.0	5.8	322.5	74.9		182.5	1.4			145.0	7.7
	125.0	5.8	137.5	3.6	325.0	98.2		190.0	6.5			152.5	3.1
	127.5	7.5	170.0	6.5	362.5	35.8		192.5	0.1			192.5	1.0
	132.5	3.6	192.5	10.5	365.0	4.5		497.5	0.7			215.0	0.7

Table C.15: The first 10 unstable circular frequencies (left column) and distributed growth rate, $\tilde{\omega}_i$, (right column) tabulated for the asymmetric harmonic Beltramian spectrum for multidirectional flow number, m (Fig. 6.32).

$\kappa = 0.1$							$\kappa = 0.01$						
$m =$	0		2		4		$m =$	0		2		4	
	105.0	0.9	87.5	1.2	45.0	1.7		90.0	0.6	62.5	2.7	77.5	4.0
	112.5	3.4	112.5	2.0	60.0	2.9		92.5	1.7	87.5	0.6	120.0	3.2
	115.0	3.1	115.0	6.5	62.5	6.1		100.0	1.7	100.0	1.2	122.5	3.2
	120.0	5.1	135.0	5.3	65.0	3.8		102.5	0.6	107.5	2.4	125.0	14.0
	127.5	5.2	137.5	4.0	67.5	0.1		135.0	0.2	110.0	7.0	127.5	14.1
	130.0	0.2	142.5	2.9	105.0	1.5		152.5	0.2	112.5	6.6	142.5	7.1
	135.0	13.7	157.5	1.6	107.5	7.5		182.5	1.0	115.0	0.5	172.5	9.9
	137.5	9.4	192.5	21.1	110.0	0.9		187.5	0.9	147.5	0.4	180.0	1.9
	145.0	8.2	195.0	33.7	117.5	19.9		190.0	0.8	160.0	2.5	182.5	12.3
	147.5	3.1	207.5	20.1	120.0	7.0		535.0	0.4	162.5	3.1	185.0	29.6

Vita

Joshua Will Batterson was born on April 11th, 1983 to Richard and Mary Batterson of Huntington, Indiana. He began his undergraduate studies at Trine University (formerly Tri-State University) in mechanical engineering in the Fall of 2001. In 2005 he completed his B.S. in Mechanical Engineering at Tri-State and began pursuing his graduate education at the University of Tennessee Space Institute under the tutelage of Dr. Joseph Majdalani. Josh received his Master of Science in Aerospace Engineering from the University of Tennessee, Knoxville in November of 2007 after completing a thesis on boundary layers in bidirectional vortex flowfields. His doctorate considered an up-and-coming approach to hydrodynamic stability in which he successfully developed general codes for three-dimensional flow analysis.

During his graduate studies, Josh filled many service rolls. He transitioned from student senator to VP of Records and Finance to student body President. During his time with the student government, he was involved with various leadership and representative roles throughout the UT system. He also was awarded the privilege of hosting the AIAA Region II student conference in 2009. Through his efforts, campus wide participation in AIAA increased. His rising position among his peers secured him an invitation to address members of congress in Washington D.C. on the merits of a well funded space program in the areas of technology development, education, and international commerce.

Josh's academic accomplishments included several conference publications in the field of vortex technologies and hydrodynamic instability in propulsive applications. Lastly, he has

made advances in asymptotic mathematics including solutions of transcendental equations and singular perturbations.