SANDIA REPORT

SAND2000-0221
Unlimited Release
Printed February 2000

Bursting Frequency Prediction in Turbulent Boundary Layers

William W. Liou and Yichung Fang

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

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (703) 605-6000
Web site: http://www.ntis.gov/ordering.htm

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

SAND2000-0221
Unlimited Release
Printed February 2000

Bursting Frequency Prediction in Turbulent Boundary Layers

William W. Liou and Yichung Fang
Department of Mechanical and Aeronautical Engineering
Western Michigan University
Kalamazoo, MI 49008

Abstract

The frequencies of the bursting events associated with the streamwise coherent structures of spatially developing incompressible turbulent boundary layers were predicted using global numerical solutions of the Orr–Sommerfeld and the vertical vorticity equations of hydrodynamic stability problems. The structures were modeled as wavelike disturbances associated with the turbulent mean flow. The global method developed here involves the use of second and fourth order accurate finite difference formulae for the differential equations as well as the boundary conditions. An automated prediction tool, **BURFIT**, was developed. The predicted resonance frequencies were found to agree very well with previous results using a local shooting technique and measured data.

Acknowledgement

Roy S. Baty was the technical monitor for this code development project.

Table of Contents

(this page intentionally left blank)

Nomenclature

A_v coefficient in equation 7

B_v coefficient in equation 7

B_η coefficient in equation 8

$\mathbf{C}$ coefficient matrices in equation 10

D $\frac{d}{dt}$

$\mathbf{D}$ lambda matrix

F_i averaged turbulence quantity

L length scale

Re Reynolds number

U mean velocity in the x–direction

c wavespeed .

i index or $\sqrt{-1}$

m transformation metric $\frac{d\bar{y}}{dy}$

t time

x x coordinate

y y coordinate

z z coordinate

U_∞ free stream velocity

$\overline{f}_i$ turbulence quantity

f_i' background fluctuation

f_i wavelike component

u_τ wall frictional velocity

$\hat{v}$ mode shape for the wavelike y–component of velocity

$\mathbf{v}$ solution vector in equation 9

$\overline{y}$ transformed y coordinate

y^+ $\frac{yu_\tau}{\nu}$

α wavenumber in the x–direction

β wavenumber in the z–direction

δ boundary layer thickness

δ^* boundary layer displacement thickness

$\hat{\eta}$ mode shape for the x–component of vorticity

ν kinematic viscosity

ω frequency

ω^+ $\frac{\omega\nu}{u_\tau^2}$

(this page intentionally left blank)

1.0 Introduction

Many experimental results on incompressible and compressible turbulent boundary layers have indicated the existence of coherent structures in such flows. The quasi-deterministic occurrence of large–scale organized structures is collectively called the bursting process. The bursting process is believed to play a dominant role in the development of turbulent boundary layers.

The bursting process is associated with the appearance of counter–rotating spanwise rolls of vortical structures, Figure 1. Experiments by Morrison & Kronauer (1969) showed that the statistically dominant streamwise fluctuations of the streamwise vortical structures exhibited wavelike characteristics, suggesting that a hydrodynamic wave description for the streamwise structures is applicable.

Based on a weakly nonlinear theory, Jang *et al.*(1986) proposed that resonance could occur for certain damped three-dimensional modes when the eigenmodes of the Orr–Sommerfeld solution corresponded to that associated with the vertical vorticity equation. As a result, they showed that for incompressible turbulent boundary layers, the secondary mean flow induced by these resonant fundamental modes contained streamwise vortical structures. The shape of the predicted structures and the spacing of the accompanying low–speed streaks are comparable with experimental observation.

Because of the nature of the numerical integration scheme used in Jang *et al.* (1986), some knowledge of the eigenvalues is required *a priori* in order for the numerical solution to be successful. Since this information is not readily available beyond a few simple profiles for the mean quantities, it severely limits the use of the direct–resonance method in simple flow cases. Furthermore, the eigenvalue spectra of the Orr–Sommerfeld and the vertical vorticity equations contain many other eigenmodes. It is possible that the eigenmodes not considered in Jang *et al.* (1986) might also excite resonance. These issues may become a major concern when the flow speed increases, and effects of compressibility are included.

In addition, the Orr–Sommerfeld and the vertical vorticity equations yield stiff systems of ordinary differential equations. During the numerical integration of a stiff system, numerical errors associated with one solution may contaminate the other and lose their linear independence. Extra care, such as the use of a re–orthonormalization procedure, is required to keep the solution independent. In this research we implemented a modern global numerical scheme for the stability problem. A global method solves the equations using a global approximation of the solution. The global solution method does not require a

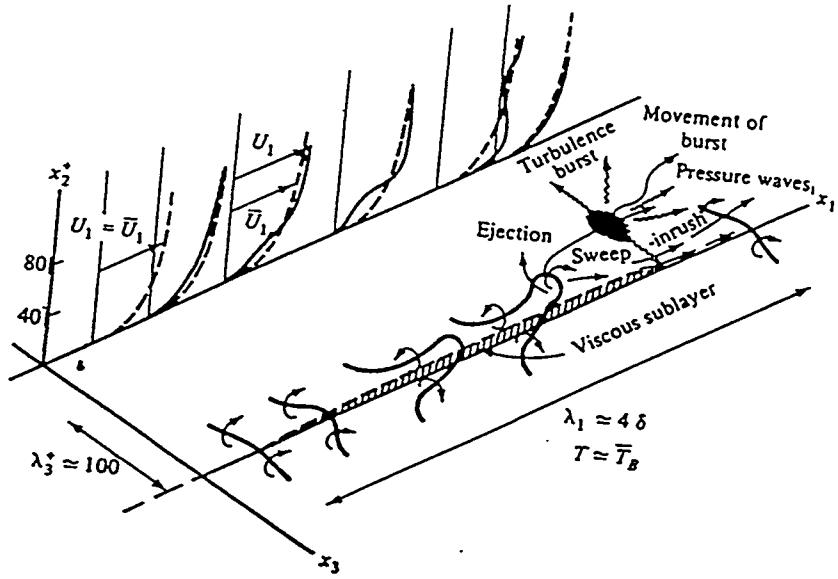

Figure 1. A sketch of the bursting event. Cantwell (1981).

re–orthonormalization procedure and is ideal for stiff systems such as the Orr-Sommerfeld and the vorticity equations, Liou and Morris (1992), Baty and Morris (1995), Bridges and Morris (1984).

The global method provides a description of the entire eigenvalue spectrum of the stability problem without using any prior knowledge of the eigenvalues, as is required by the traditional shooting procedure. As such, all possible bursting frequencies are likely to be identified automatically without artificial intervention. This capability allows an efficient model of the bursting frequencies in incompressible as well as compressible turbulent boundary layers.

In this research, second and fourth order accurate finite difference formulae have been used in approximating the Orr-Sommerfeld equation, the vertical vorticity equation, and their boundary conditions.

In the following, the derivation and the solution of the equations are described. The results are presented in section 4.

2.0 Modeling

Turbulence quantities, $\overline{f_i}$, are decomposed into three components:

$$\overline{f_i} = F_i + f_i + f_i', \tag{1}$$

where F_i represents a long–time average of $\overline{f_i}$, f_i the wavelike component of $\overline{f_i}$, and f_i' the background fluctuation. Substituting equation (1) into the Navier–Stokes equations, followed by a linearization of the disturbance quantities, the equations governing the mode shape of the vertical velocity, $\hat{v}$, the Orr–Sommerfeld equation, and the homogeneous vertical vorticity, $\hat{\eta}$, equation can be found:

$$\left[\mathrm{i}(\alpha U - \omega)(D^2 - \alpha^2 - \beta^2) - \mathrm{i}\alpha D^2 U - \frac{1}{R}(D^2 - \alpha^2 - \beta^2)^2\right]\hat{v} = 0 \tag{2}$$

$$\left[\mathrm{i}(\alpha U - \omega) - \frac{1}{R}(D^2 - \alpha^2 - \beta^2)\right]\hat{\eta} = 0 \tag{3}$$

Equations (2) and (3) have been derived by assuming a normal mode solution for the wavelike disturbances, f_i, i.e.,

$$f_i = \hat{f}_i \mathrm{e}^{\mathrm{i}(\alpha x + \beta z - \omega t)}. \tag{4}$$

In equations (2) and (3), U represents the streamwise mean velocity, and $D \equiv \mathrm{d}/\mathrm{d}y$.

Equations (2) and (3) govern the mode shape of wavelike disturbances associated with the mean quantities in terms of the streamwise and spanwise wavenumbers, α and β; the wave frequency, ω; and the flow Reynolds number, $R(= \frac{U_\infty L}{\nu})$. Liou and Morris (1992) used a similar wavelike description for the coherent structures in free shear layers and successfully calculated the evolution of the time–dependent turbulent motion at the large scale. Their analysis, which requires a weakly nonlinear solution for the wave amplitude, will not be performed here for the bounded shear flows. In this study the bursting frequencies of the streamwise coherent structures in turbulent boundary layers are sought using the direct–resonance model. The condition for direct resonance can be written as

$$c^{OS}(\alpha, \beta, R) = c^{VV}(\alpha, \beta, R) \tag{5}$$

where c^{OS} and c^{VV} represent the phase velocity, ω/α, associated with the Orr–Sommerfeld and the vertical vorticity eigenvalue problems.

The boundary conditions for $\hat{v}$ and $\hat{\eta}$ are

$$\hat{v} = D\hat{v} = \hat{\eta} = 0 \quad \text{at} \quad y = 0,\ \infty. \tag{6}$$

In the following section the numerical solution of equations (2), (3), (5), and (6) are described.

3.0 Numerical Solutions

3.1 Mean Flow

The local mean velocity in the streamwise direction, U, for the turbulent boundary layer over a flat plate can be obtained from the solutions of either the boundary layer equations or the Navier–Stokes equations. We found the resulting eigenvalue problems sensitive to the profile shapes of U and D^2U. For the results presented, the U and D^2U profiles were obtained by using dense grids ($\approx$ 1000) in a boundary–layer equation solver to retain a higher order of fidelity of the velocity as well as its second order derivative. A mixing-length turbulence model was used for the turbulent eddy–viscosity.

3.2 Wavelike Structure

To resolve the near–wall behavior of the boundary layer, an algebraic stretching of the grid was used beginning from the wall. In the transformed coordinate, $\bar{y}$, the equations are

$$m(m(m(m\hat{v}')')')' + A_v m(m\hat{v}')' + B_v\hat{v} = 0 \tag{7}$$

$$m(m\hat{\eta}')' + B_\eta\hat{\eta} = 0, \tag{8}$$

where

$$A_v = -2(\alpha^2+\beta^2) - \mathrm{i}R(\alpha U - \omega)$$

$$B_v = (\alpha^2+\beta^2)^2 + \mathrm{i}R(\alpha^2+\beta^2)(\alpha U - \omega) + \mathrm{i}R\alpha D^2U$$

$$B_\eta = -\mathrm{i}R(\alpha U - \omega) - (\alpha^2+\beta^2)$$

and

$$()' \equiv \frac{\mathrm{d}}{\mathrm{d}\bar{y}}$$

$\bar{y}$ denotes the transformed coordinate. The global solution method developed in this research for the Orr–Sommerfeld and the vertical vorticity equations involves the use of second and fourth order accurate finite difference formulae for the equations and the boundary conditions. The second order formulae are widely available. The fourth order formulae used here are listed in the Appendix. The resulting homogeneous systems of equations form eigenvalue

problems, for both the Orr–Sommerfeld and the vertical vorticity equations, nonlinear in the parameter, α. For the Orr–Sommerfeld equation the system can be written as

$$\mathbf{D}_4(\alpha)\mathbf{v} = 0 \tag{9}$$

The matrix, $\mathbf{D}_4$, is a lambda matrix of degree four, Lancaster (1966), and can be expressed as a scalar polynomial with matrix coefficients:

$$\mathbf{D}_4(\alpha) = \mathbf{C_0}\alpha^4 + \mathbf{C_1}\alpha^3 + \mathbf{C_2}\alpha^2 + \mathbf{C_3}\alpha + \mathbf{C_4}. \tag{10}$$

With the inclusion of the boundary conditions, the matrices $\mathbf{C's}$ are square matrices of order n, which represents the number of grid point in $\bar{y}$. A linear companion matrix method, Bridges and Morris (1984), was used to linearize the lambda matrix. The resulting general eigenvalue problem becomes

$$\left\{ \begin{pmatrix} -\mathbf{C_1} & -\mathbf{C_2} & -\mathbf{C_3} & -\mathbf{C_4} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{pmatrix} - \alpha \begin{pmatrix} \mathbf{C_0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \right\} \begin{pmatrix} \alpha^3\hat{v} \\ \alpha^2\hat{v} \\ \alpha\hat{v} \\ \hat{v} \end{pmatrix} = \mathbf{0}. \tag{11}$$

Equation (11) can be further transformed to an algebraic eigenvalue problem seeking the eigenvalues of matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} -\mathbf{C_0^{-1}C_1} & -\mathbf{C_0^{-1}C_2} & -\mathbf{C_0^{-1}C_3} & -\mathbf{C_0^{-1}C_4} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{pmatrix}. \tag{12}$$

The eigenvalues may be obtained by using QR or QZ algorithms. The details of the formulation and the application of the Linear Companion Matrix method can be found in Bridges and Morris (1984) and Liou and Morris (1995).

The resonance in the stability problem occurs when there is a set of parameters (α, β, ω) for which the solutions of the Orr–Sommerfeld and the vertical vorticity equations exist for a given mean velocity distribution and a Reynolds number. To locate the resonance mode, we choose to solve the following equations:

$$\alpha_r^{OS}(\beta,\omega) - \alpha_r^{VV}(\beta,\omega) = 0 \tag{13a}$$

$$\alpha_i^{OS}(\beta,\omega) - \alpha_i^{VV}(\beta,\omega) = 0 \tag{13b}$$

A subroutine in the IMSL package called NEQBF has been used to solve the system of equations.

4.0 Results

In this section the solutions of the resonance stability problem obtained by using the high order finite difference global method are described.

A polynomial type of distribution was first used as the mean velocity profile. The profile

$$U(\frac{y}{\delta}) = 2(\frac{y}{\delta}) - 2(\frac{y}{\delta})^3 + (\frac{y}{\delta})^4$$

was often used as an approximation to the streamwise velocity for flat–plate boundary layers. δ represents the boundary–layer thickness. The eigenvalues obtained by using the present global method were compared with those obtained by using the local shooting method. The comparisons are shown in Table 1.

Table 1: Comparison of the calculated eigenvalues, α.

Re	ω	Equations	Present	Shooting
8,000	0.2354	Orr–Sommerfeld	(0.780864,-0.013533)	(0.781148,-0.01356)
10,000	0.1202	vertical vorticity	(0.41512,0.40891)	(0.415119,0.408911)

Figure 2 shows the calculated complex phase velocity for a plane mode, and $Re(\frac{U_\infty \delta}{\nu}) = 8,000$ and 10,000 for the vertical vorticity equation. The frequencies are 0.0122 and 0.1202, respectively. An analytical form of the continuous spectrum was also included for comparison, Grosch and Salwen (1978). For $Re = 10,000$, the two discrete Tollmien–Schlichiting instability modes can be clearly identified. For $Re = 8,000$, the discrete spectrum appearers close to the continuous spectrum. For both cases the continuous spectrum are well–resolved.

Numerical experiments were conducted to examine the effects of the order of the finite difference approximation, the number of grid points, and the location of the outer boundary of the computed domain in y. Second–, as well as fourth–, order approximations were applied to the differential equations and the boundary conditions. Figure 3 shows a result of the calculated complex phase velocity for a plane mode for the Orr–Sommerfeld equation using various order of finite differencing, numbers of grid point, and $(\frac{y}{\delta})_{max}$. Figure 3 includes the Tollmien–Schlichiting instability modes and the continuous spectra for the approximate laminar boundary layer profile. The symbols represent the results for the various calculated cases denoted by a-b-c, where [a] denotes the order of accuracy, [b] the number of discretized points used, and [c] the far field boundary distance to the wall. The first, second, and the third number in [a] represent the order of accuracy for the derivatives in the Orr–Sommerfeld

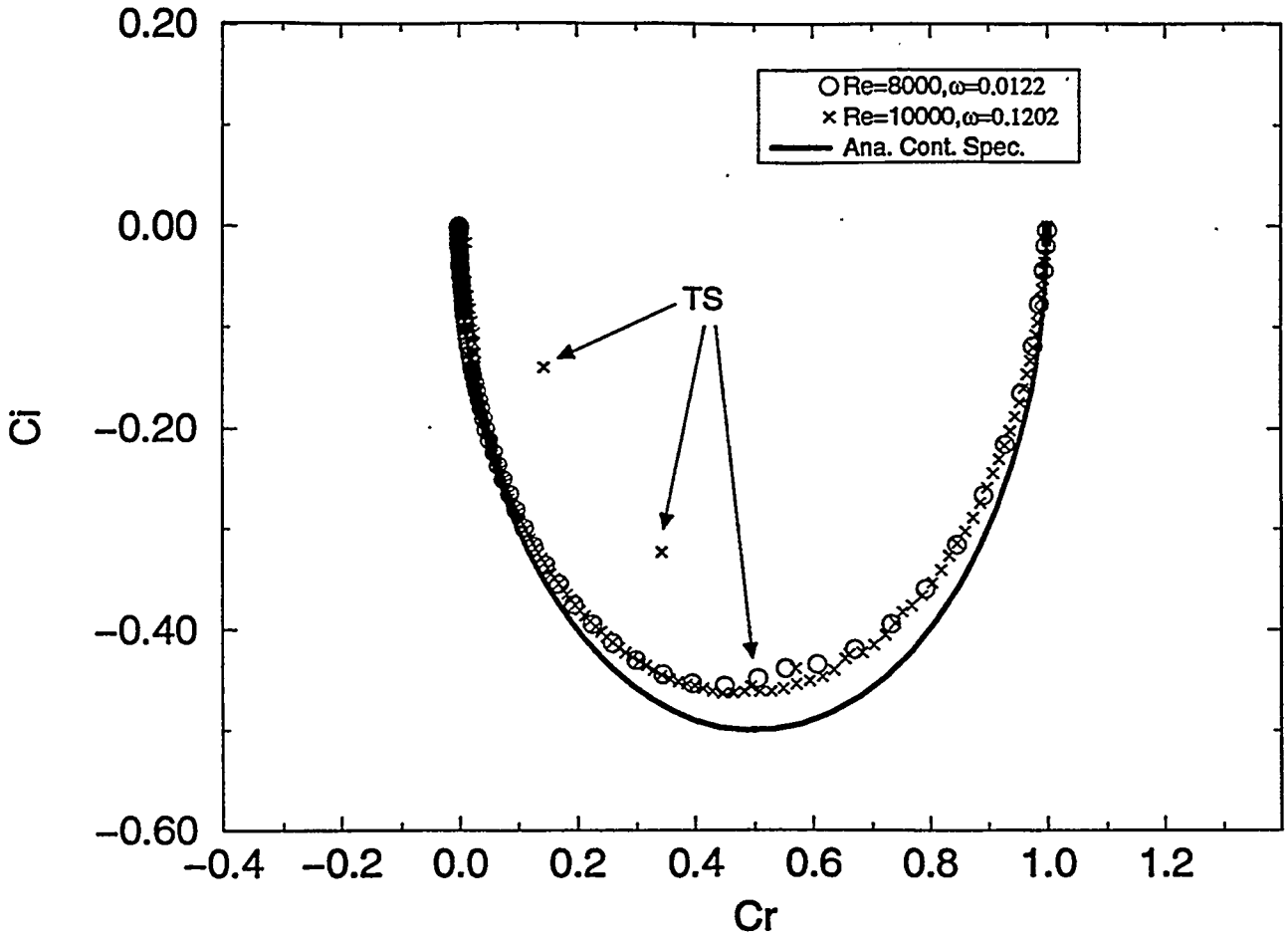

Figure 2. Complex wave speed. vertical vorticity equation.

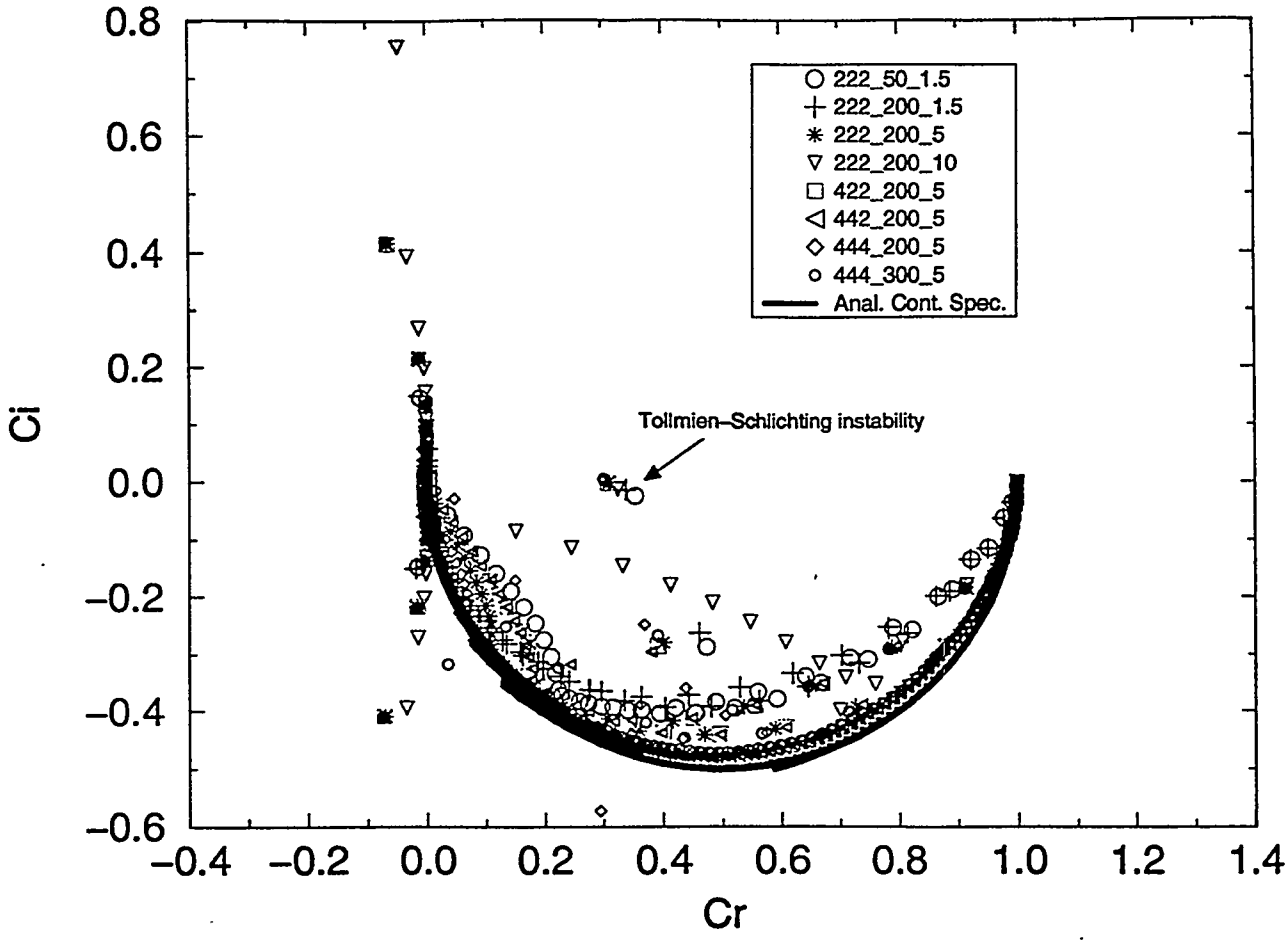

Figure 3. Comparison of eigenvalue spectra. Orr–Sommerfeld equation.

equation, for the far field boundary condition, and for the wall boundary condition, respectively. The agreement between the computed and the analytical continuous spectrum improves with the increasing number of grid nodes. It also appears that increasing the order of accuracy of the discretization enhances the agreement. The results shown in the following were obtained by using the fourth–order formulae and a grid of 200 nodes.

To calculate the eigenvalues of the direct–resonance problems associated with a turbulent flat plate boundary layer, the mean streamwise velocity and its second derivative distribution were needed. A boundary–layer equation solver using the Prandtl's mixing length model with the van Driest damping function was developed. The calculated turbulent mean velocity distribution for $Re = 1,000$ is shown in Figure 4. The results compared well with the log law–of–the–wall in the log layer of the boundary layer. The number of grid points is 998. The second derivative of the mean velocity, D^2U, was obtained using a fourth–order accurate finite difference formula. Figure 5 shows the D^2U distribution. For laminar boundary layers, the solutions of the Orr–Sommerfeld and vertical vorticity equations are known to be sensitive to the input velocity and its second derivative. We found that this sensitivity of the Orr–Sommerfeld and vertical vorticity equations to the input U and D^2U remains for the current problem involving turbulent boundary layers. As can be seen in figures 4 and 5, the U and D^2U profiles vary significantly over a small distance in the near–wall region of the flow. It is necessary that the multiple length scales of turbulent boundary layers be captured properly in the Orr–Sommerfeld and vertical vorticity problems. Algebraically stretched grids were used to ensure an appropriate resolution of the different regions. Comprehensive numerical experiments showed that a parameter set of $y_1^+ = 1$ and $(\frac{y}{\delta^*})_{max} = y_l = 50$ gives the best results in terms of both the discrete, as well as the continuous, spectra of the Orr–Sommerfeld and vertical vorticity equations for a wide range of Reynolds numbers. δ^* denotes the displacement thickness and $()_1$ the first grid point away from the wall. Figure 6 shows some results of the numerical experiments. Figure 6 shows the complex phase velocity spectrum of the vertical vorticity equation for $\omega = 2$; $\beta = 10$; $y_l = 50$; and $y_1^+ = 0.001, 0.01, 0.1, 1.0, 5.0$. Except for $y_1^+ = 5$, the computed discrete modes agree well. There is a more significant separation between the discrete and the continuous spectra for $y_1^+ = 1$ than for the other values. This separation between the discrete and the continuous modes was used as a criterion to identify the discrete modes from the continuous modes in an automated procedure of bursting frequency prediction. A calculation with $y_1^+ = 1$ and $y_l = 90$ shows no changes in either the discrete or the

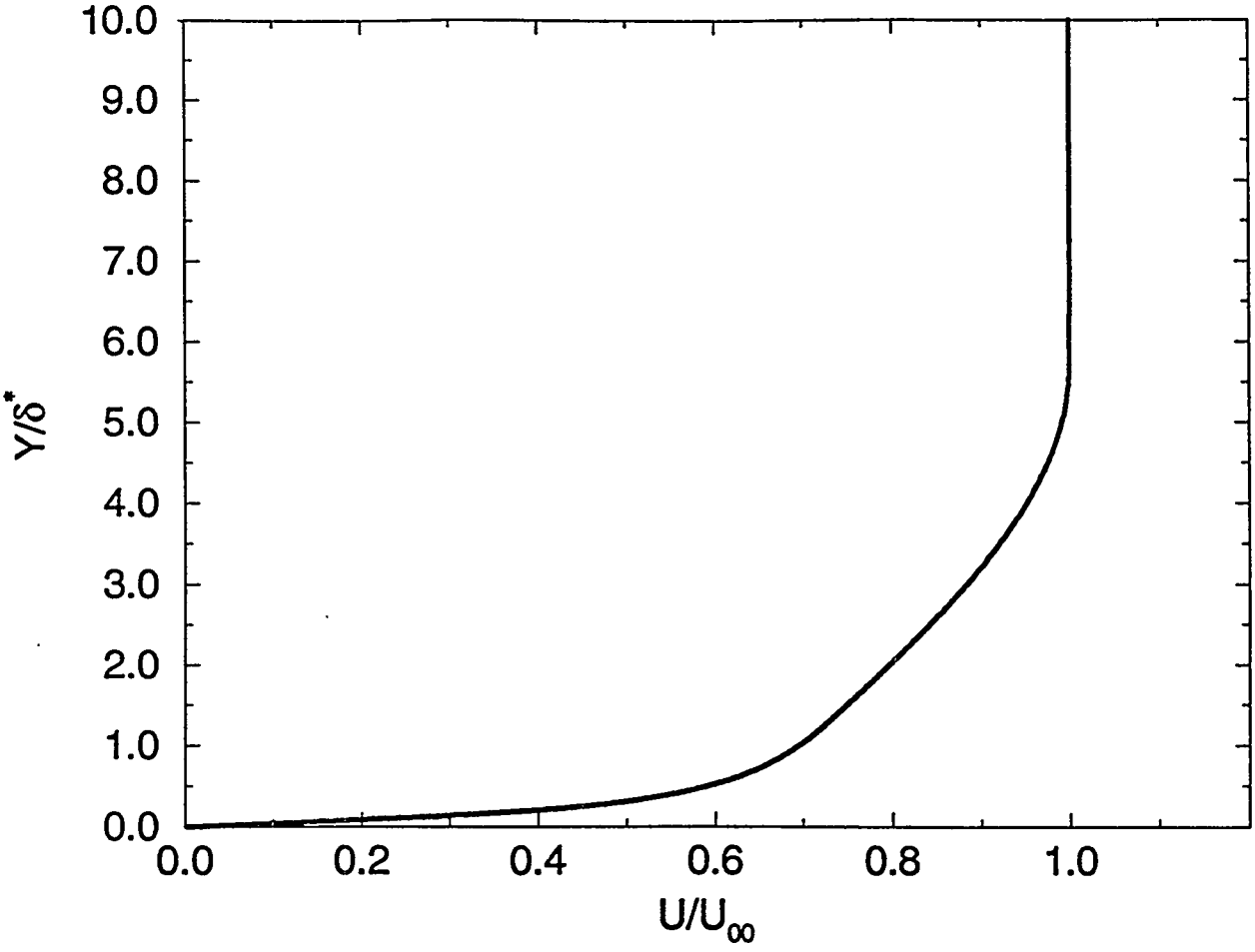

Figure 4. Streamwise mean velocity.

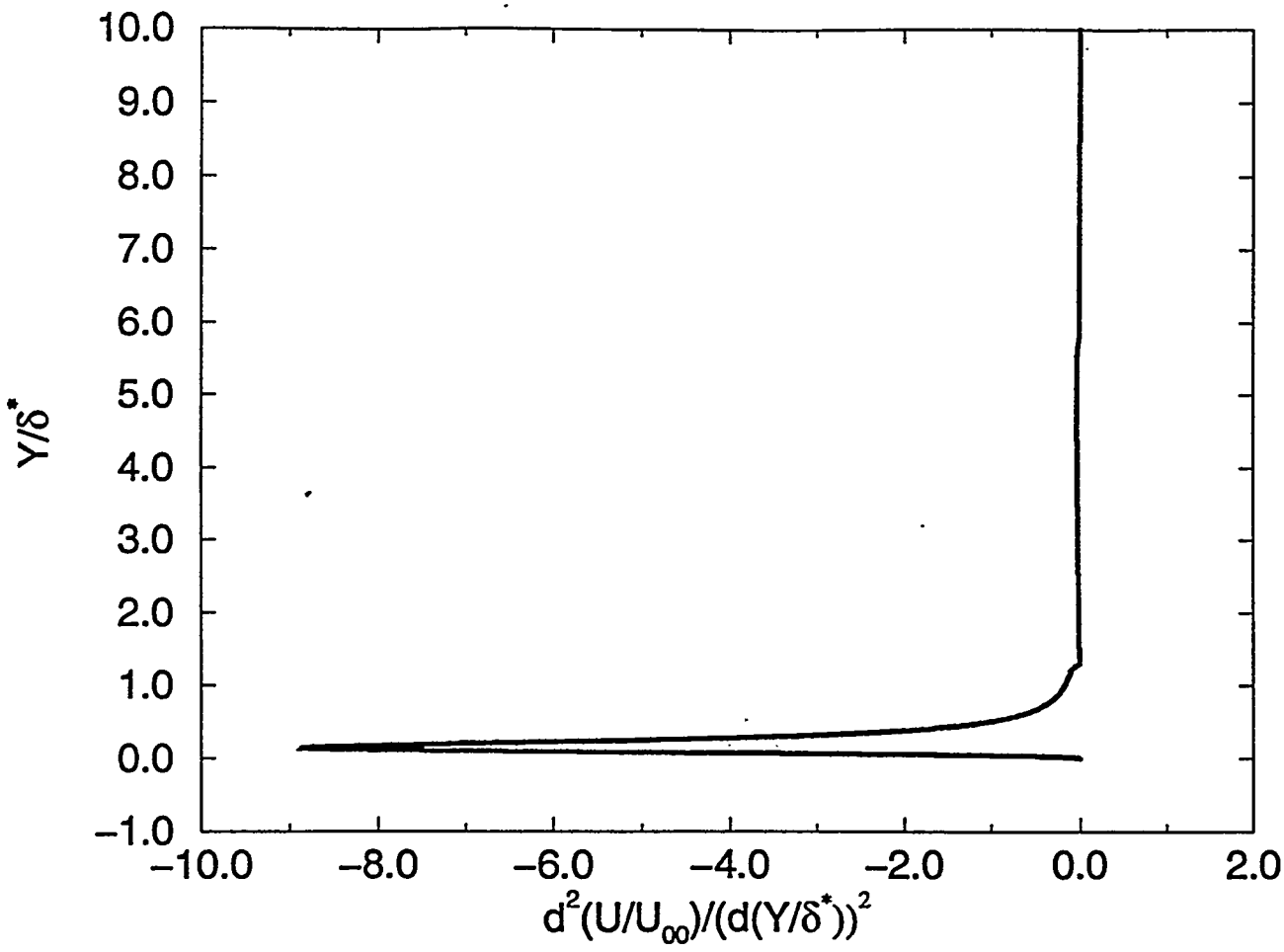

Figure 5. Second order derivative of the mean velocity.

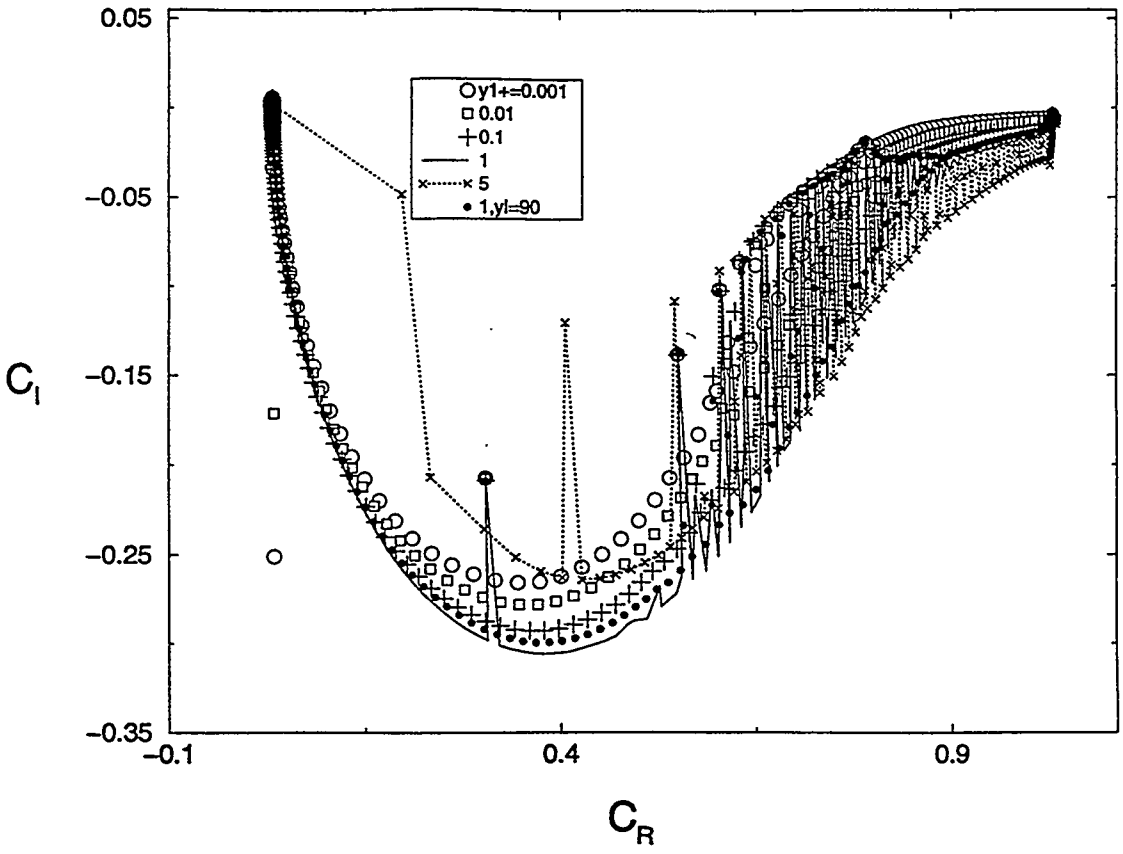

Figure 6. Results of parametric studies.

continuous spectra. Figure 7 shows the eigenvalue spectrum for the Orr–Sommerfeld equation for a turbulent boundary layer for $Re(\frac{U_\infty \delta^*}{\nu}) = 1,000$.

As was stated earlier, based on the direct–resonance theory, an automated procedure has been developed for the prediction of the bursting frequencies associated with the streamwise structures in turbulent boundary layers. To identify a resonance mode, the Orr–Sommerfeld equation and the homogeneous vertical vorticity equation were first solved. Resonance occurs when the eigenvalues of the Orr–Sommerfeld solution correspond to that associated with the vertical vorticity equation. The resonance condition has been written in the form of a system of two equations, equation (13), nonlinear in their parameters, ω and β. The resonance mode is identified when a solution of equation (13) is found. The solution of equation (13) involves an iteration process. The search for resonance mode is complete when the solution of equation (13) is obtained. The procedure for searching the resonance mode has been automated. The automated search procedure was implemented in a FORTRAN software called **BURFIT** (**BUR**sting **F**requency prediction in **I**ncompressible **T**urbulent boundary layers).

Figures 8, 9, and 10 show the results of using BURFIT for a turbulent boundary layer of $Re = 1,000$. Figures 8 and 9 show the convergent history for the complex α and c, respectively. The search process was terminated when the right–hand side of equation (13), denoted by d_R and d_I, have reduced to the order of –4. Figure 10 shows the evolution of d_R and d_I. The predicted resonant frequency is ω^+=0.0962, which compares well with that of Jang *et al.*(1986), of 0.09, calculated by using a shooting method and the measured data of Morrison and Kronauer (1969).

The eigenvalue spectra for the Orr–Sommerfeld and vertical vorticity equations at the resonance condition for $Re = 8,000$ are shown in figure 11. When the resonance condition was met, there was apparently a matching of not only the discrete mode but also the continuous modes.

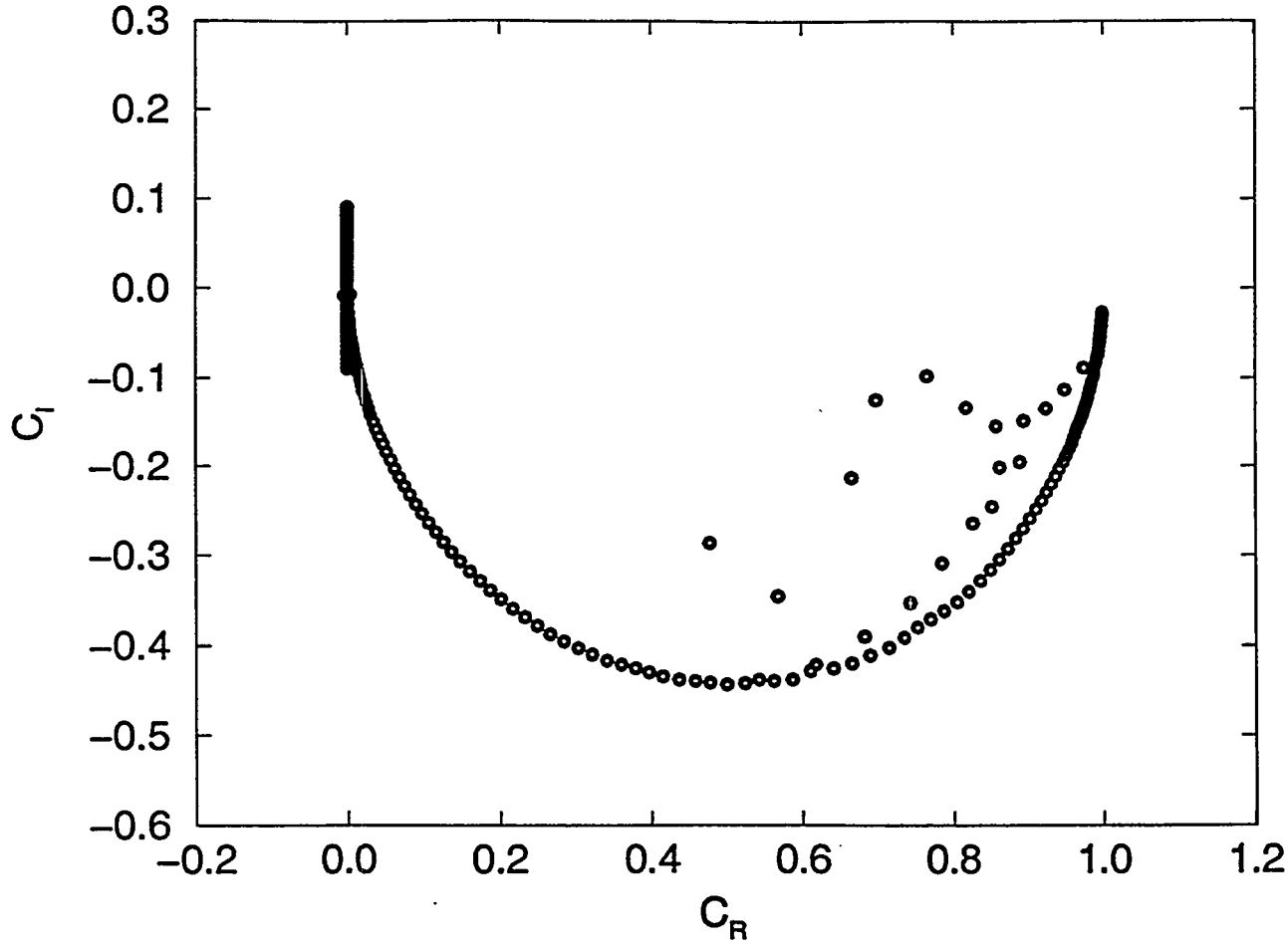

Figure 7. Eigenvalue spectrum. Turbulent. Orr–Sommerfeld equation

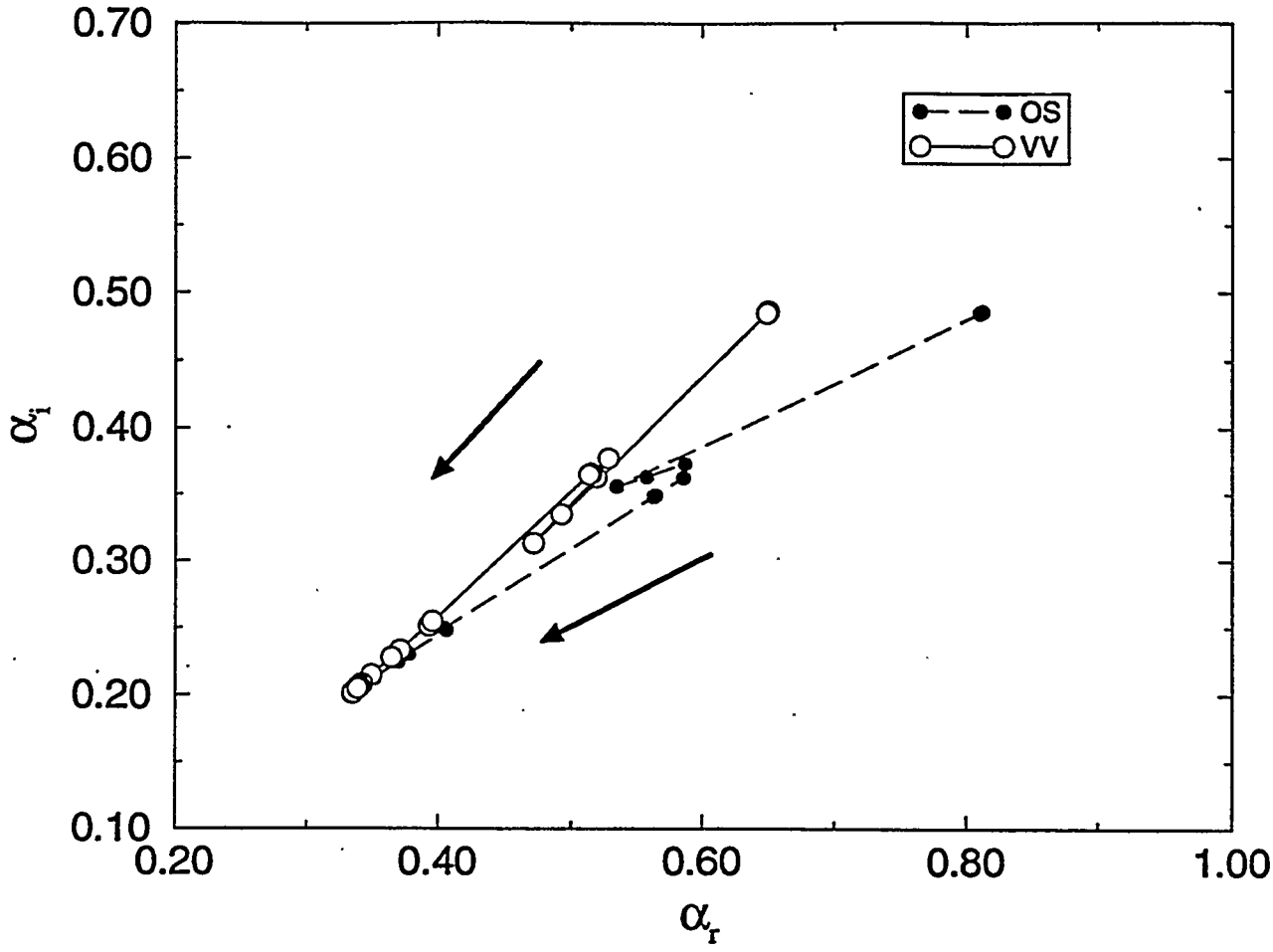

Figure 8. Convergent history of α.

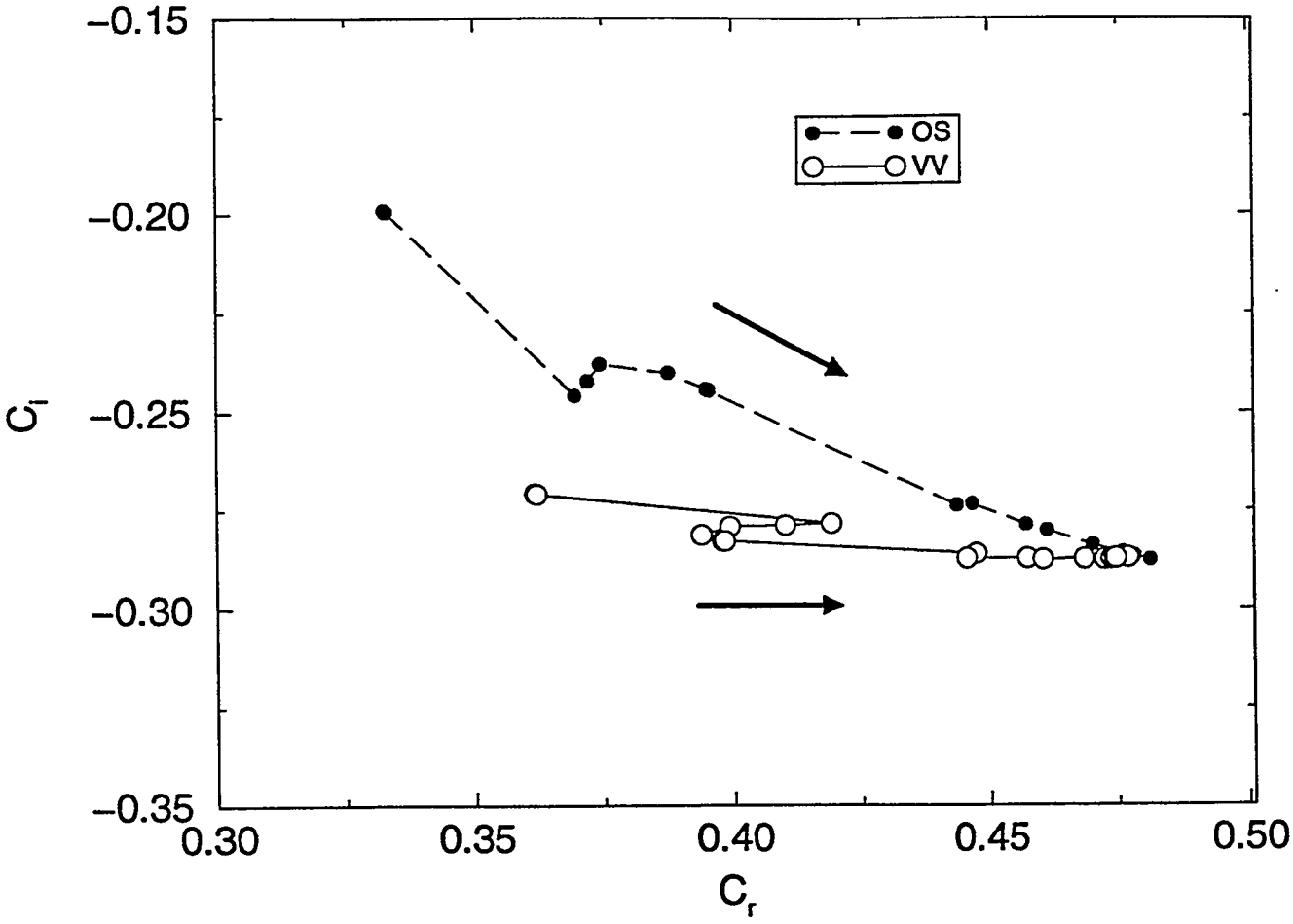

Figure 9. Convergent history of c.

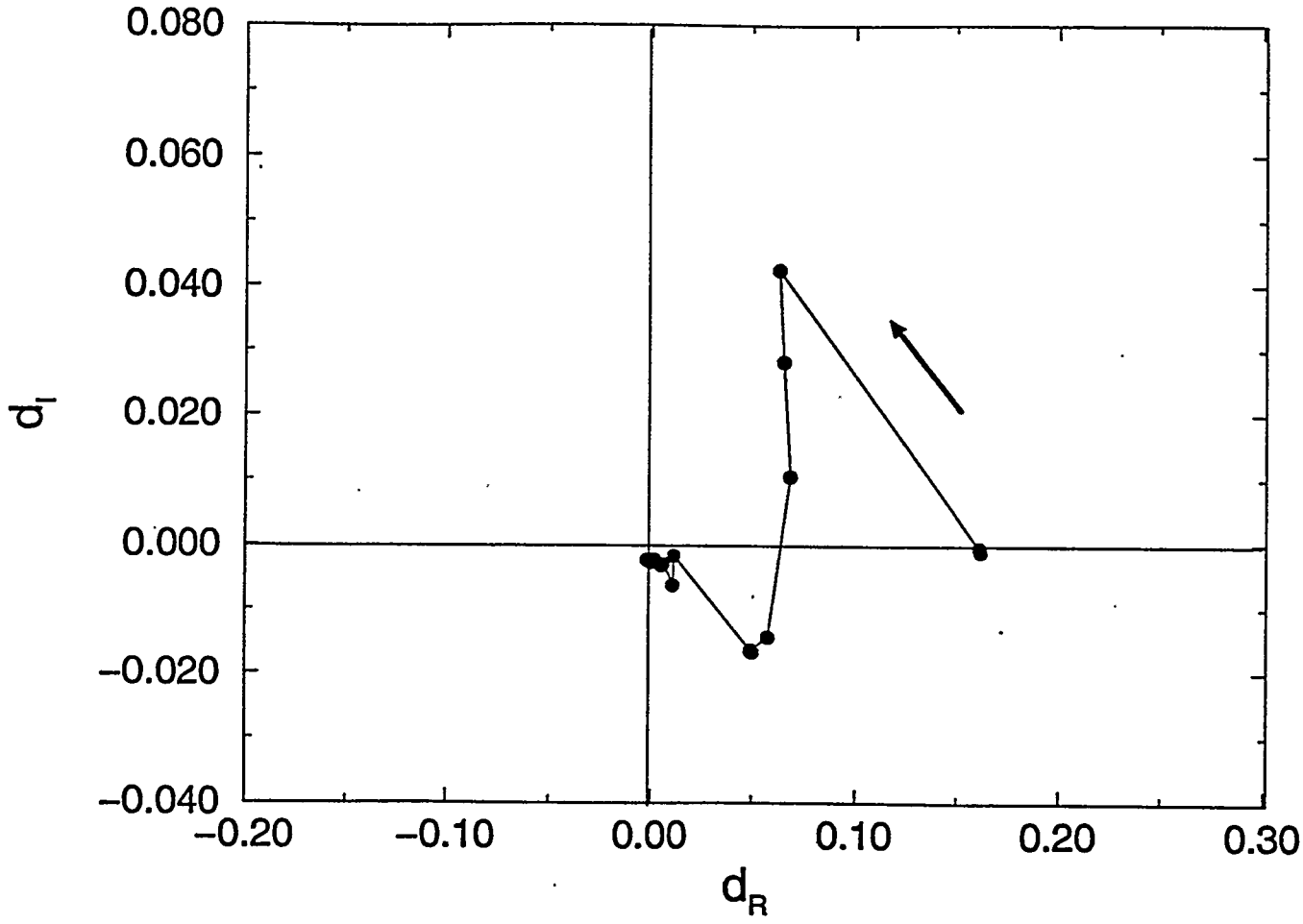

Figure 10. Convergent history BURFIT

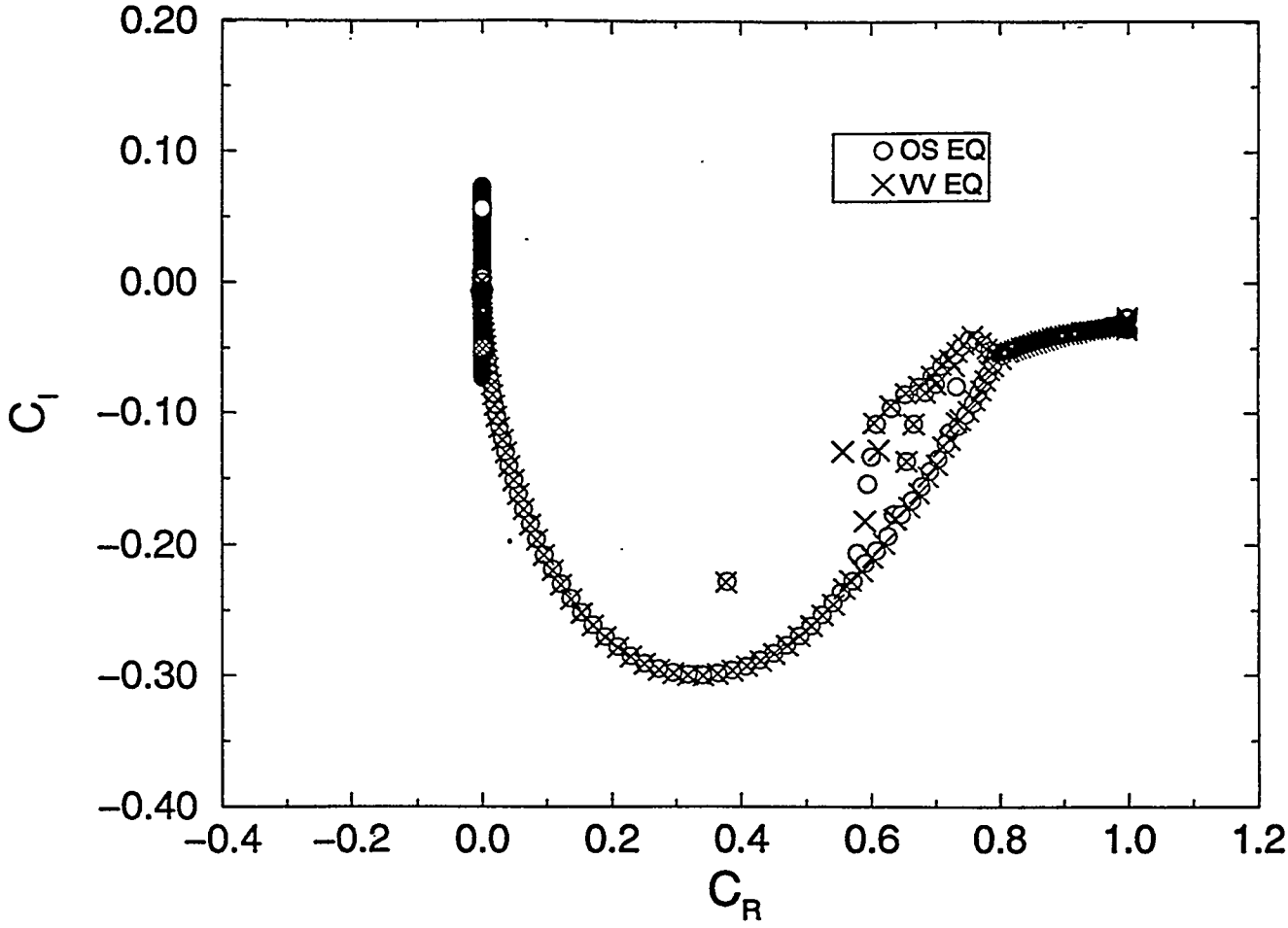

Figure 11. Eigenvalue spectrum at resonance. Re=8,000. Orr–Sommerfeld equation

5.0 Concluding Remarks

In this study a global solution method has been successfully developed to predict the bursting frequencies associated with the coherent streamwise structures in incompressible turbulent boundary layers. The prediction was based on the direct resonance model. The prediction tool developed, BURFIT, has been automated, and no artificial intervention is necessary other than assigning the starting values. In the present study the predicted bursting frequencies have been found to agree very well with previous numerical calculations and experimental data for incompressible turbulent boundary layers over a flat plate. The development of a global mechanism is also necessary when a direct resonance theory, perhaps similar to the one implemented here for incompressible flow, is extended to the prediction of the bursting frequencies of high-speed turbulent boundary layers. There can be more than one discrete mode associated with the bursting events when the effects of compressibility are considered.

References

T.J. Bridges and P.J. Morris, *J. Computational Physics*, **55**, 437 (1984).

C.E. Grosch and H. Salwen, *JFM*, **87**, 33 (1978).

P.S. Jang, D.J. Benny, and R.L. Gran, *JFM*, **169**, 109 (1986).

P. Lancaster, *Lambda Matrices and Vibrating Systems*, Permagon, Oxford, 1966.

W.W. Liou and P.J. Morris, *International Journal of Numerical Methods in Fluids*, **15**, 1407 (1992).

W.W. Liou and P.J. Morris, *Physics of Fluids*, 4, 2798 (1992).

W.R.B. Morrison and R.E. Kronauer, *JFM*, **39**, 117 (1969).

R.S. Baty and P.J. Morris, *International Journal of Numerical Methods in Fluids*, **21**, 763 (1995).

Appendix A

The fourth order, $O(\Delta^4)$, finite difference formulae used in the current study are given below. All formulae are given for the derivatives at a point denoted by $()_0$ and its neighboring points in decreasing order, $()_{-1}$, $()_{-2}$,..., and in increasing order, $()_{+1}$, $()_{+2}$,..., etc.. Δ denotes the uniform grid spacing.

$$f_0' = \frac{f_{-2} - 8f_{-1} + 8f_1 - f_2}{12\Delta}$$

$$f_0' = \frac{-3f_{-1} - 10f_0 + 18f_1 - 6f_2 + f_3}{12\Delta}$$

$$f_0' = \frac{3f_1 + 10f_0 - 18f_{-1} + 6f_{-2} - f_{-3}}{12\Delta}$$

$$f_0' = \frac{-25f_0 + 48f_1 - 36f_2 + 16f_3 - 3f_4}{12\Delta}$$

$$f_0' = \frac{25f_0 - 48f_{-1} + 36f_{-2} - 16f_{-3} + 3f_{-4}}{12\Delta}$$

$$f_0'' = \frac{-2f_{-2} + 16f_{-1} - 30f_0 + 16f_1 - f_2}{12\Delta^2}$$

$$f_0'' = \frac{10f_{-1} - 15f_0 - 4f_1 + 14f_2 - 6f_3 + f_4}{12\Delta^2}$$

$$f_0'' = \frac{10f_1 - 15f_0 - 4f_{-1} + 14f_{-2} - 6f_{-3} + f_{-4}}{12\Delta^2}$$

$$f_0''' = \frac{f_{-3} - 8f_{-2} + 13f_{-1} - 13f_1 + 8f_2 - f_3}{8\Delta^3}$$

$$f_0''' = \frac{-f_{-2} - 8f_{-1}35f_0 - 48f_1 + 29f_2 - 8f_3 + f_4}{8\Delta^3}$$

$$f_0''' = \frac{f_2 + 8f_1 - 35f_0 + 48f_{-1} - 29f_{-2} + 8f_{-3} - f_{-4}}{8\Delta^3}$$

$$f_0'''' = \frac{-f_{-3} + 12f_{-2} - 39f_{-1} + 56f_0 - 39f_1 + 12f_2 - f_3}{6\Delta^4}$$

$$f_0'''' = \frac{20f_{-2} - 55f_{-1} + 155f_1 - 220f_2 + 135f_3 - 40f_4 + 5f_5}{30\Delta^4}$$

$$f_0'''' = \frac{20f_2 - 55f_1 + 155f_{-1} - 220f_{-2} + 135f_{-3} - 40f_{-4} + 5f_{-5}}{30\Delta^4}$$

Appendix B

A listing of the code **BURFIT** is included in this appendix. Two subroutines that were used in the code were taken out due to copyright. They are called LUBKSB and LUDCMP from *Numerical Recipes.*

```
c
c**********************************************************************c
c                                                                      c
c                        BURFIT                                        c
c                                                                      c
c       William W. Liou                                                c
c       Western Michigan University                                    c
c       July 31, 1999                                                  c
c       For Sandia National Laboratories                               c
c                                                                      c
c       The package is a software tool for the solution of the         c
c       linearized model, based on finite difference global numerical  c
c       method, for bursting frequency prediction of incompressible    c
c       turbulent boundary layer over flat plates.                     c
c                                                                      c
c**********************************************************************c
c**********************************************************************c
c                                                                      c
c       Locate Direct Resonance using IMSL subroutine NEQBF            c
c                                                                      c
c**********************************************************************c
c
        integer iparam(6)
        real fcn, fscale(2),fvec(2),rparam(5),x(2),xguess(2),xscale(2)
        external fcn,neqbf
        data xscale/2*1./, fscale/2*1./
c
        xguess(1)=0.3076573
        xguess(2)=2.33344
c
c ipapam array controls iteration parameters
c
        iparam(1)=0
c
        call neqbf (fcn,2,xguess,xscale,fscale,iparam,rparam,x,fvec)
        print *,'   Ha! Ha! Ha! Converged!  omega =  ',
     &          x(1),'  beta=  ',x(2)
        stop
        end
c
        subroutine fcn(ndumy,x,rf)
c
c**********************************************************************c
c                                                                      c
c       7/8/1999 for SNL                                               c
c       V-V Eq. and OS EQ.                                             c
c       length scale: delta(u=0.99U)                                   c
c       velocity scale: free stream U                                  c
c       nmax : total number of points and the size of the matrices     c
c       Fourth order accurate for all derivatives                      c
c       Transformed domain                                             c
c                                                                      c
c**********************************************************************c
c
c       dimnesions, common blocks, variables types....
c
        parameter (nmax=201)
        implicit complex (c)
        real x(2),rf(2)
        complex   c0(nmax,nmax),c1(nmax,nmax),c2(nmax,nmax)
     &           ,c3(nmax,nmax),c4(nmax,nmax),ca(nmax*4,nmax*4)
     &           ,b0(nmax,nmax),b1(nmax,nmax),b2(nmax,nmax)
     &           ,b3(nmax,nmax),b4(nmax,nmax)
     &           ,cunit(nmax,nmax),cwork(nmax,nmax),alpha(nmax*4)
     &           ,alphal(nmax),vl(nmax,nmax),vr(nmax,nmax),work(nmax*2)
     &           ,f,g,cwavespd(2)
```

```
      real      m,mp,mpp,mppp
      dimension y(nmax),umean(nmax),umeanpp(nmax)
     &          ,rwork(nmax*8),indx(nmax),disc(nmax*4,nmax*4)
     &          ,rm(nmax),rmp(nmax),rmpp(nmax),rmppp(nmax)
      namelist /input/ neqq,omegaq,betaq,yplus1,yl
      character jobvl,jobvr
c
      omega=x(1)
      beta=x(2)
c
c Set dimensions of parameter window
c
      if(omega.gt.0.6.or.beta.gt.7.5)then
      print *,'********  OUTOF BOUND',omega,beta
      stop
      endif
c
c Do OS(neq=1) and VV(2) eqns.
c
      do 100 neq=1,2
c
c Parameters
c
      nmax4=nmax*4
      nmax2=nmax*2
      open(1,status='unknown',file='ov.i')
      read(1,input)
      close(1)
c     write(*,*) neqq, omegaq, betaq, yplus1, yl
c
      open(1,status='unknown',file='meaninfo')
      read(1,*) redelst
      read(1,*) y1,u1,pp1
      close(1)
      re=redelst
c Find du/dy|wall
      wshear=u1/y1
c
c initializaiton
c
      czero=(0.,0.)
      cone=(1.,0.)
      ci=(0.,1.)
      do i=1,nmax
      do j=1,nmax
         c0(i,j)=czero
         c1(i,j)=czero
         c2(i,j)=czero
         c3(i,j)=czero
         c4(i,j)=czero
         cunit(i,j)=czero
         cwork(i,j)=czero
      enddo
      enddo
c
c Find coordinates
c
     njm1=nmax
     njm=njm1-1
c
     dely=yplus1/sqrt(wshear*re)
c
     ray1=1.
     dy1=y1/float(njm1-1)
     ray=1.01
c
```

```
      ddel=ray-ray1
      do icount=1,1000
         toy=0.
         do j=2,njm
            toy=toy+ray**(j-2)
         enddo
         toy=toy+(njm1-njm)*ray**(njm-2)
         dy=yl/toy
         ddel=ddel*(dy1-dely)/(dy1-dy)*0.1
         ray1=ray
         dy1=dy
         ray=ray1+ddel
         if(abs(ddel).le.1.e-8)go to 5
      enddo
      print *,' ray did not converge !!! Latest are ',ray,ddel
 5    continue
      y(1)=0.
      do j=2,njm
         y(j)=y(j-1)+dy*ray**(j-2)
      enddo
      do 1305 j=njm+1,njm1
         y(j)=y(j-1)+dy*ray**(njm-2)
      enddo
c
c Find m', m'', m'''
c     m
         i=1
         rm(i)=12./(-25.*y(i  )+48.*y(i+1)
     &             -36.*y(i+2)+16.*y(i+3)-3.*y(i+4))
         i=2
         rm(i)=12./(-3.*y(i-1)-10.*y(i  )+18.*y(i+1)
     &             -6.*y(i+2)+    y(i+3))
         do i=3,nmax-2
            rm(i)=12./(y(i-2)-8.*y(i-1)+8.*y(i+1)-y(i+2))
         enddo
         i=nmax-1
         rm(i)=12./(- 1.*y(i-3)+6.*y(i-2)-18.*y(i-1)
     &             +10.*y(i  )+3.*y(i+1))
         i=nmax
         rm(i)=12./(  3.*y(i-4)-16.*y(i-3)+36.*y(i-2)
     &             -48.*y(i-1)+25.*y(i  ))
c     m'
         i=2
         rmp(i)=(-3.*rm(i-1)-10.*rm(i)+18.*rm(i+1)-6.*rm(i+2)
     &          +rm(i+3))/12.
         do i=3,nmax-2
            rmp(i)=(rm(i-2)-8.*rm(i-1)+8.*rm(i+1)-rm(i+2))/12.
         enddo
         i=nmax-1
         rmp(i)=(- 1.*rm(i-3)+6.*rm(i-2)-18.*rm(i-1)
     &          +10.*rm(i  )+3.*rm(i+1))/12.
c     m''
         do i=3,nmax-2
            rmpp(i)=(- 1.*rm(i-2)+16.*rm(i-1)-30.*rm(i)
     &              +16.*rm(i+1)- 1.*rm(i+2))/12.
         enddo
c     m'''
         i=3
         rmppp(i)=(- 1.*rm(i-2)-8.*rm(i-1)+35.*rm(i  )-48.*rm(i+1)
     &            +29.*rm(i+2)-8.*rm(i+3)+ 1.*rm(i+4))/8.
         do i=4,nmax-3
            rmppp(i)=(    rm(i-3)-8.*rm(i-2)+13.*rm(i-1)-13.*rm(i+1)
     &               +8.*rm(i+2)-1.*rm(i+3))/8.
         enddo
         i=nmax-2
         rmppp(i)=(-    rm(i-4)+8.*rm(i-3)-29.*rm(i-2)+48.*rm(i-1)
```

```
     &            -35.*rm(i  )+8.*rm(i+1)+    rm(i+2))/8.
c
c Find U
c
        call mean(nmax,y,umean,umeanpp)
        write(14,9030) (y(i),umean(i),umeanpp(i),rm(i),rmp(i)
     &                 ,rmpp(i),rmppp(i),i=1,nmax)
c
c-----------------------------------------------------------------------
c
c Select OS (neq=1) or VV (neq=2) Eq.
c
        if( neq.eq.1 ) then
c
c
c O-S equation
c Begin here
c
        nmaxeq=nmax4
c
c Matrix entries
c Zero, 1st and 2nd derivatives
c
        cargu=2.*beta**2-ci*re*omega
        do 15 i=3,nmax-2
c
           ubar=umean(i)
           ubarpp=umeanpp(i)
           m=rm(i)
           mp=rmp(i)
           mpp=rmpp(i)
           mppp=rmppp(i)
           c0v    =cone
           c1v    =ci*re*ubar
           c2v    =cargu
           c2vp   =-2.*m*mp
           c2vpp  =-2.*m**2
           c3v    =ci*re*beta**2*ubar+ci*re*ubarpp
           c3vp   =-ci*re*ubar*m*mp
           c3vpp  =-ci*re*ubar*m**2
           c4v    =beta**4-ci*re*omega*beta**2
           c4vp   =m*mp**3+4.*m**2*mp*mpp+m**3*mppp+m*mp*(-cargu)
           c4vpp  =7.*m**2*mp**2+4.*m**3*mpp+m**2*(-cargu)
c   Zero
           c0(i,i)=c0v
           c1(i,i)=c1v
           c2(i,i)=c2v
           c3(i,i)=c3v
           c4(i,i)=c4v
c   first
           c2(i,i-2)= (1./12.)*c2vp
           c2(i,i-1)=-(8./12.)*c2vp
           c2(i,i+1)= (8./12.)*c2vp
           c2(i,i+2)=-(1./12.)*c2vp
c
           c3(i,i-2)= (1./12.)*c3vp
           c3(i,i-1)=-(8./12.)*c3vp
           c3(i,i+1)= (8./12.)*c3vp
           c3(i,i+2)=-(1./12.)*c3vp
c
           c4(i,i-2)= (1./12.)*c4vp
           c4(i,i-1)=-(8./12.)*c4vp
           c4(i,i+1)= (8./12.)*c4vp
           c4(i,i+2)=-(1./12.)*c4vp
c   second
           c2(i,i-2)=-( 1./12.)*c2vpp+c2(i,i-2)
```

```
           c2(i,i-1)= (16./12.)*c2vpp+c2(i,i-1)
           c2(i,i  )=-(30./12.)*c2vpp+c2(i,i  )
           c2(i,i+1)= (16./12.)*c2vpp+c2(i,i+1)
           c2(i,i+2)=-( 1./12.)*c2vpp+c2(i,i+2)
c
           c3(i,i-2)=-( 1./12.)*c3vpp+c3(i,i-2)
           c3(i,i-1)= (16./12.)*c3vpp+c3(i,i-1)
           c3(i,i  )=-(30./12.)*c3vpp+c3(i,i  )
           c3(i,i+1)= (16./12.)*c3vpp+c3(i,i+1)
           c3(i,i+2)=-( 1./12.)*c3vpp+c3(i,i+2)
c
           c4(i,i-2)=-( 1./12.)*c4vpp+c4(i,i-2)
           c4(i,i-1)= (16./12.)*c4vpp+c4(i,i-1)
           c4(i,i  )=-(30./12.)*c4vpp+c4(i,i  )
           c4(i,i+1)= (16./12.)*c4vpp+c4(i,i+1)
           c4(i,i+2)=-( 1./12.)*c4vpp+c4(i,i+2)
c
 15     continue
c
c 3rd and 4th derivatives in c4 only
c
        do 16 i=4,nmax-3
           m=rm(i)
           mp=rmp(i)
           c4vppp =6.*m**3*mp
           c4vpppp=m**4
c    3rd
           c4(i,i-3)= ( 1./8.)*c4vppp
           c4(i,i-2)=-( 8./8.)*c4vppp+c4(i,i-2)
           c4(i,i-1)= (13./8.)*c4vppp+c4(i,i-1)
           c4(i,i+1)=-(13./8.)*c4vppp+c4(i,i+1)
           c4(i,i+2)= ( 8./8.)*c4vppp+c4(i,i+2)
           c4(i,i+3)=-( 1./8.)*c4vppp
c    4th
           c4(i,i-3)=-(1. /6.)*c4vpppp+c4(i,i-3)
           c4(i,i-2)= (12./6.)*c4vpppp+c4(i,i-2)
           c4(i,i-1)=-(39./6.)*c4vpppp+c4(i,i-1)
           c4(i,i)  = (56./6.)*c4vpppp+c4(i,i)
           c4(i,i+1)=-(39./6.)*c4vpppp+c4(i,i+1)
           c4(i,i+2)= (12./6.)*c4vpppp+c4(i,i+2)
           c4(i,i+3)=-(1. /6.)*c4vpppp+c4(i,i+3)
 16     continue
c i=3
        i=3
        m=rm(i)
        mp=rmp(i)
        c4vppp =6.*m**3*mp
        c4vpppp=m**4
c 3rd
        c4(i,i-1)=-(15./8.)*c4vppp+c4(i,i-1)
        c4(i,i  )= (56./8.)*c4vppp+c4(i,i  )
        c4(i,i+1)=-(83./8.)*c4vppp+c4(i,i+1)
        c4(i,i+2)= (64./8.)*c4vppp+c4(i,i+2)
        c4(i,i+3)=-(29./8.)*c4vppp+c4(i,i+3)
        c4(i,i+4)= ( 8./8.)*c4vppp+c4(i,i+4)
        c4(i,i+5)=-( 1./8.)*c4vppp+c4(i,i+5)
c 4th
        c4(i,i-2)= ( 20./30.)*c4vpppp+c4(i,i-2)
        c4(i,i-1)=-( 55./30.)*c4vpppp+c4(i,i-1)
        c4(i,i+1)= (155./30.)*c4vpppp+c4(i,i+1)
        c4(i,i+2)=-(220./30.)*c4vpppp+c4(i,i+2)
        c4(i,i+3)= (135./30.)*c4vpppp+c4(i,i+3)
        c4(i,i+4)=-( 40./30.)*c4vpppp+c4(i,i+4)
        c4(i,i+5)= (  5./30.)*c4vpppp+c4(i,i+5)
c i=nmax-2
        i=nmax-2
```

```
        m=rm(i)
        mp=rmp(i)
        c4vppp =6.*m**3*mp
        c4vpppp=m**4
c 3rd
        c4(i,i-5)= ( 1./8.)*c4vppp+c4(i,i-5)
        c4(i,i-4)=-( 8./8.)*c4vppp+c4(i,i-4)
        c4(i,i-3)= (29./8.)*c4vppp+c4(i,i-3)
        c4(i,i-2)=-(64./8.)*c4vppp+c4(i,i-2)
        c4(i,i-1)= (83./8.)*c4vppp+c4(i,i-1)
        c4(i,i  )=-(56./8.)*c4vppp+c4(i,i  )
        c4(i,i+1)= (15./8.)*c4vppp+c4(i,i+1)
c 4th
        c4(i,i+2)= ( 20./30.)*c4vpppp+c4(i,i+2)
        c4(i,i+1)=-( 55./30.)*c4vpppp+c4(i,i+1)
        c4(i,i-1)= (155./30.)*c4vpppp+c4(i,i-1)
        c4(i,i-2)=-(220./30.)*c4vpppp+c4(i,i-2)
        c4(i,i-3)= (135./30.)*c4vpppp+c4(i,i-3)
        c4(i,i-4)=-( 40./30.)*c4vpppp+c4(i,i-4)
        c4(i,i-5)= ( 5./30.)*c4vpppp+c4(i,i-5)
c
c BC's
c v=0
c
          c4(1,1)=cone
          c4(nmax,nmax)=cone
c
c v'=0
c Zero first derivative at boundary (fourth order accurate)
c
          c4(2,1)=-25.
          c4(2,2)= 48.
          c4(2,3)=-36.
          c4(2,4)= 16.
          c4(2,5)=-3.
          c4(nmax-1,nmax-4)= 3.
          c4(nmax-1,nmax-3)=-16.
          c4(nmax-1,nmax-2)= 36.
          c4(nmax-1,nmax-1)=-48.
          c4(nmax-1,nmax  )= 25.
c
c Transformation
c
          f=2.*cone
          g=2.*cone
          do i=1,nmax
          do j=1,nmax
             b0(i,j)= c0(i,j)*(f**4)
     &               +c1(i,j)*(f**3)
     &               +c2(i,j)*(f**2)
     &               +c3(i,j)*(f)
     &               +c4(i,j)*(1.)
             b1(i,j)= c0(i,j)*(-4.*f**3*g)
     &               +c1(i,j)*(-3.*f**2*g+f**3)
     &               +c2(i,j)*(-2.*f*g+2.*f**2)
     &               +c3(i,j)*(-g+3.*f)
     &               +c4(i,j)*(4.)
             b2(i,j)= c0(i,j)*(6.*f**2*g**2)
     &               +c1(i,j)*(3.*f*g**2-3.*f**2*g)
     &               +c2(i,j)*(g**2-4.*f*g+f**2)
     &               +c3(i,j)*(-3.*g+3.*f)
     &               +c4(i,j)*(6.)
             b3(i,j)= c0(i,j)*(-4.*f*g**3)
     &               +c1(i,j)*(-g**3+3.*f*g**2)
     &               +c2(i,j)*(2.*g**2-2.*f*g)
     &               +c3(i,j)*(-3.*g+f)
```

```
     &                +c4(i,j)*(4.)
              b4(i,j)= c0(i,j)*(g**4)
     &                +c1(i,j)*(-g**3)
     &                +c2(i,j)*(g**2)
     &                +c3(i,j)*(-g)
     &                +c4(i,j)*(1.)
         enddo
         enddo
c
c End of OS Eq.
c-----------------------------------------------------------------------
c
         else if( neq.eq.2 )then
c
c V-V equation.
c Begin here
c
c
         nmaxeq=nmax2
c
c Matrix entries
c Zero, 1st and 2nd derivatives
c
         do i=3,nmax-2
            ubar=umean(i)
            c0v  =-cone
            c1v  =-ci*re*ubar
            c2v  = ci*re*omega-beta**2
            c2vp = rm(i)*rmp(i)
            c2vpp= rm(i)**2
c
            c2(i,i-2)= (1./12.)*c2vp-(1. /12.)*c2vpp
            c2(i,i-1)=-(8./12.)*c2vp+(16./12.)*c2vpp
c
            c0(i,i)=c0v
            c1(i,i)=c1v
            c2(i,i)=c2v-(30./12.)*c2vpp
c
            c2(i,i+1)= (8./12.)*c2vp+(16./12.)*c2vpp
            c2(i,i+2)=-(1./12.)*c2vp-(1. /12.)*c2vpp
         enddo
c i=2
         i=2
         ubar=umean(i)
         c0v  =-cone
         c1v  =-ci*re*ubar
         c2v  = ci*re*omega-beta**2
         c2vp = rm(i)*rmp(i)
         c2vpp= rm(i)**2
c
         c0(i,i)=c0v
         c1(i,i)=c1v
         c2(i,i)=c2v
c
         c2(i,i-1)=-( 3./12.)*c2vp+(10./12.)*c2vpp
         c2(i,i  )=-(10./12.)*c2vp-(15./12.)*c2vpp+c2(i,i)
         c2(i,i+1)= (18./12.)*c2vp-( 4./12.)*c2vpp
         c2(i,i+2)=-( 6./12.)*c2vp+(14./12.)*c2vpp
         c2(i,i+3)= ( 1./12.)*c2vp-( 6./12.)*c2vpp
         c2(i,i+4)= ( 1./12.)*c2vpp
c i=nmax-1
         i=nmax-1
         ubar=umean(i)
         c0v  =-cone
         c1v  =-ci*re*ubar
         c2v  = ci*re*omega-beta**2
```

```
        c2vp = rm(i)*rmp(i)
        c2vpp= rm(i)**2
c
        c0(i,i)=c0v
        c1(i,i)=c1v
        c2(i,i)=c2v
c
        c2(i,i-4)= ( 1./12.)*c2vpp
        c2(i,i-3)=-( 1./12.)*c2vp-( 6./12.)*c2vpp
        c2(i,i-2)= ( 6./12.)*c2vp+(14./12.)*c2vpp
        c2(i,i-1)=-(18./12.)*c2vp-( 4./12.)*c2vpp
        c2(i,i  )= (10./12.)*c2vp-(15./12.)*c2vpp+c2(i,i)
        c2(i,i+1)= ( 3./12.)*c2vp+(10./12.)*c2vpp
c
c BCs
c v=0
        c2(1,1)=cone
        c2(nmax,nmax)=cone
c
c Transformation
c
        f=cone
        g=cone
        do i=1,nmax
        do j=1,nmax
           b0(i,j)= c0(i,j)*(f**2)
     &             +c1(i,j)*(f)
     &             +c2(i,j)
           b1(i,j)= c0(i,j)*(-2.*f*g)
     &             +c1(i,j)*(f-g)
     &             +c2(i,j)*(2.)
           b2(i,j)= c0(i,j)*(g**2)
     &             +c1(i,j)*(-g)
     &             +c2(i,j)*(1.)
           b3(i,j)= czero
           b4(i,j)= czero
        enddo
        enddo
c
c-----------------------------------------------------------------------
        else
        print *,'  ===== NO SUCH CASE ====='
        print *,'  ===== BING!  IS NEQ > 2 ?  ====='
        end if
c-----------------------------------------------------------------------
c
c end of equation selection
c
c
c Form companion matrix in ca
c
        call lcmm( nmax, b0, b1, b2, b3, b4, ca, cunit, cwork, indx )
c
c Get eigenvalues using IMSL
c
        do i=1,nmaxeq
           alpha(i)=czero
        enddo
        call EVLCG(nmaxeq,ca,nmax4,alpha)
        do i=1,nmaxeq
           calphatemp=(alpha(i)*f-g)/(alpha(i)+cone)
           alpha(i)=calphatemp
        enddo
c
c Find Discrete Eigenvalue
c
```

```
        do i=1,nmaxeq
        do j=1,nmaxeq
           disc(i,j)=abs(omega/alpha(j)-omega/alpha(i))
        enddo
        enddo
        do i=1,nmaxeq
           disc(i,i)=1.e+10
        enddo
        dmax=0.
        imax=1
        do i=1,nmaxeq
           d=1.e+10
           do j=1,nmaxeq
              d=min(d,disc(i,j))
           enddo
           if( d.gt.dmax )then
               dmax=d
               imax=i
           endif
        enddo
c
        print *,'  '
        print *,'  neq,imax,alpha(imax),omega/alpha(imax)'
        write(*,*) neq,imax,alpha(imax),omega/alpha(imax)
        write(16,*)neq,imax,alpha(imax),omega/alpha(imax)
        print *,'  omega+,      alpha+'
        write(*,*) omega/wshear,alpha(imax)/sqrt(re*wshear)
c
        cwavespd(neq)=alpha(imax)
 100    continue
        rf(1)= real(cwavespd(1)-cwavespd(2))
        rf(2)=aimag(cwavespd(1)-cwavespd(2))
        print *,'  '
        print *,'  Differences'
        write(*,*)rf(1),rf(2)
        print *,'  '
        print *,'  '
        write(16,*)x(1),x(2),rf(1),rf(2)
c
 9010   format(i5,1p4e15.5)
 9020   format(10(10f6.1/)//)
 9030   format(1x,1p7e15.4)
c
        return
        end
c
c*************************************************************c
c                       LCMM                                  c
c       Linear Compaian Matrix Method                         c
c*************************************************************c
        subroutine lcmm( nmax, c0, c1, c2, c3, c4 , ca
     &                 , cunit, cwork, indx )
        complex   c0(nmax,nmax),c1(nmax,nmax),c2(nmax,nmax)
     &           ,c3(nmax,nmax),c4(nmax,nmax),ca(nmax*4,nmax*4)
     &           ,cunit(nmax,nmax),cwork(nmax,nmax)
        dimension indx(nmax)
c
c Inverse c0 and replace
c
        call minv(c0,nmax,cunit,indx)
        call meq(nmax,cunit,c0)
c
c Multiply c0inv to c1, c2, c3, and c4
c
        call mmp(nmax,c0,c1,cwork)
        call meq(nmax,cwork,c1)
```

```
        call mmp(nmax,c0,c2,cwork)
        call meq(nmax,cwork,c2)
        call mmp(nmax,c0,c3,cwork)
        call meq(nmax,cwork,c3)
        call mmp(nmax,c0,c4,cwork)
        call meq(nmax,cwork,c4)
c
c Assemble the big A matrix in ca
c
        nmax2=2*nmax
        nmax3=3*nmax
        nmax4=4*nmax
        call munit(nmax,cunit)
c
        do i=1,nmax4
        do j=1,nmax4
           ca(i,j)=(0.,0.)
        enddo
        enddo
c
        do i=1,nmax
        do j=1,nmax
           ca(i,j)=-c1(i,j)
           ca(i,j+ nmax)=-c2(i,j)
           ca(i,j+nmax2)=-c3(i,j)
           ca(i,j+nmax3)=-c4(i,j)
           ca(i+nmax,j )=cunit(i,j)
           ca(i+nmax2,j+nmax )=cunit(i,j)
           ca(i+nmax3,j+nmax2)=cunit(i,j)
        enddo
        enddo
c
        return
        end
c*************************************************************c
c                              MEAN                           c
c       Mean U and U''                                        c
c*************************************************************c
        subroutine mean(nmax,y,umean,umeanpp)
        dimension y(nmax),umean(nmax),umeanpp(nmax),
     &            yy(2000),uu(2000),pp(2000)
c
        kform=2
        if( kform .eq. 1 )then
c
c analytical form
c
        do i=1,nmax
           yi=y(i)
           if( yi.le.0.9999 )then
               umean(i)=2.*yi-2.*yi**3+yi**4
               umeanpp(i)=-12.*yi+12.*yi**2
           else
               umean(i)=1.0
               umeanpp(i)=0.0
           endif
        enddo
c
        else
c
c Discrete Form
c
c
        open(1,status='unknown',file='meaninfo')
        read(1,*) redelst
        i=0
```

```
 5      i=i+1
        read(1,*,end=10) yy(i),uu(i),pp(i)
        goto 5
c
 10     print *,' Have read   ',i,' data point for U and U'''
        close(1)
c
        is = 1
        do i=2,nmax
 15        is=is+1
           if( y(i).gt.yy(is) )goto 15
           rm=(y(i)-yy(is-1))/(yy(is)-yy(is-1))
           rp=(yy(is)-y(i))  /(yy(is)-yy(is-1))
           umean(i)  =rm*uu(is)+rp*uu(is-1)
           umeanpp(i)=rm*pp(is)+rp*pp(is-1)
           is=is-2
        enddo
  11    umean(1)=0.
        umeanpp(1)=0.
c
        endif
c
        return
        end
c*************************************************************c
c                              MUNIT                          c
c       Unit matrix                                           c
c*************************************************************c
        subroutine munit(nmax,cunit)
        complex cunit(nmax,nmax)
        do i=1,nmax
        do j=1,nmax
           cunit(i,j)=(0.0,0.0)
        enddo
        enddo
        do i=1,nmax
           cunit(i,i)=(1.0,0.0)
        enddo
        return
        end
c*************************************************************c
c                              MMP                            c
c       matrix multiplication                                 c
c*************************************************************c
        subroutine mmp(nmax,cm2,cm3,cwork)
        complex cm2(nmax,nmax),cm3(nmax,nmax),cwork(nmax,nmax)
c
        do i=1,nmax
        do j=1,nmax
           cwork(i,j)=(0.0,0.0)
        enddo
        enddo
c
        do i=1,nmax
        do j=1,nmax
           do k=1,nmax
              cwork(i,j)=cwork(i,j)+cm2(i,k)*cm3(k,j)
           enddo
        enddo
        enddo
c
        return
        end
c*************************************************************c
c                              MEQ                            c
c       Replace cm3 by cm2                                    c
```

```
c*************************************************************c
        subroutine meq(nmax,cm2,cm3)
        complex cm2(nmax,nmax),cm3(nmax,nmax)
        do i=1,nmax
        do j=1,nmax
           cm3(i,j)=cm2(i,j)
        enddo
        enddo
        return
        end
C*************************************************************C
C                              MINV                           C
c       Matrix Inversion                                      c
C*************************************************************C
        SUBROUTINE MINV(A,N,U,indx)
        COMPLEX  A(N,N),U(N,N)
        DIMENSION INDX(N)
        DO 5 J=1,N
  5     INDX(J) = 0
        CALL munit(N,U)
        CALL LUDCMP(A,N,N,INDX,D)
        DO 10 J=1,N
        CALL LUBKSB(A,N,N,INDX,U(1,J))
  10    CONTINUE
        RETURN
        END
c------------------------------  END OF Fortran File--------------
```

DISTRIBUTION

Dr. William W. Liou (20)
2065 Kohrman Hall
Department of Mechanical and Aeronautical Engineering
Western Michigan University
Kalamazoo, MI 49008

Yichung Fang (10)
2065 Kohrman Hall
Department of Mechanical and Aeronautical Engineering
Western Michigan University
Kalamazoo, MI 49008

Dr. Philip J. Morris
233 Hammond Building
Department of Aerospace Engineering
Penn State University
State College, PA 16801

Professor Eli Reshotko, Ph.D.
Case School of Engineering
Case Western Reserve University
10900 Euclid Avenue
Cleveland, OH 44106-7222

Professor Akiva Yaglom
MIT
Department of Aeronautics and Astronautics
Bldg 33-207
77 Massachusetts Ave
Cambridge, MA 02139

MS 0828 T. C. Bickel, 9101
MS 0828 R. K. Thomas, 9102
MS 0836 C. W. Peterson, 9104
MS 0835 S. N. Kempka, 9111
MS 0825 W. H. Rultedge, 9115

MS 0825 F. G. Blottner, 9115
MS 0825 D. L. Potter, 9115 (20)
MS 0828 M. Pilch, 9133
MS 0828 W. L. Oberkampf, 9133
MS 0828 R. S. Baty, 9133 (15)
MS 9018 Central Technical Files, 8940-2
MS 0899 Technical Library, 4916 (2)
MS 0612 Review & Approval Desk (For DOE/OSTI), 4912