ME 634
Turbulent Flows

Problem Set 1:

Derivation and Numerical Analysis of the Orr-Sommerfeld Equation

Bryan Kaiser
3/26/13

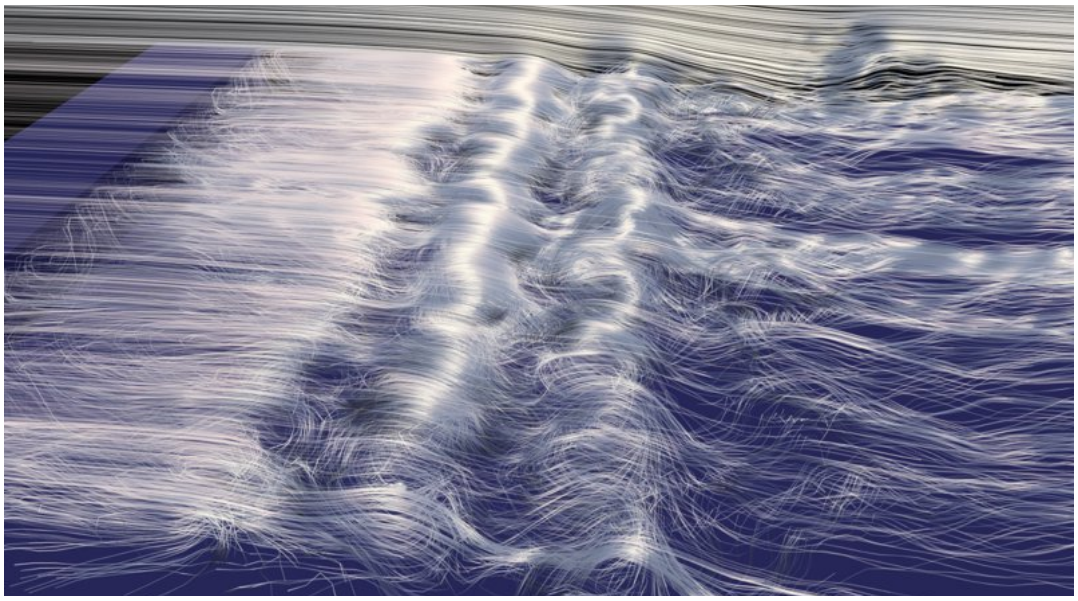

Table of Contents

Part (A)

Derive the 2D small-disturbance equations from the Navier-Stokes equations. Carefully explain all assumptions. Explain why the 3D small disturbance equations have the same form as the 2D equations.

Solution:

Laminar shear layers become unstable at a Reynolds number that depends on the shape of the velocity profile.[1] Boundary layers on stream-wise flat surfaces become unstable at a Reynolds number of approximately 1600,[1] but may remain stable up to a Reynolds number of 10,000 in carefully controlled experiments. The most unstable type of disturbance for a flat surface is a sinusoidal fluctuation of the stream-wise and cross-stream velocity components within the flow, moving downstream as a traveling wave at a speed somewhere near the average speed of the shear layer fluid and having a wavelength of 5-10 the boundary layer thickness.[1] If the velocity fluctuations grow to approximately 1/10 the free stream velocity over a flat plate, secondary instabilities are generated and the flow may become unstable in the spanwise direction.[1]

Linear stability theory (LST) is the simplest theory available to quantitatively describe the effect of instabilities on the flow. LST provides an upper bound to the critical Reynolds number for a given flow (Re_L, shown below in Figure 1), above which all unstable flows will transition into turbulence.[2] For Re $<Re_E$ or region I the flow is monotonically stable, for $Re_E<$ Re $<Re_G$ or region II the flow is globally stable, and for $Re_G<$ Re $<Re_L$ or region III there exists at least one infinitesimal disturbance that is unstable.[2] Beyond Re_L or region IV infinitesimal disturbances amplify into sustained turbulence.[2]

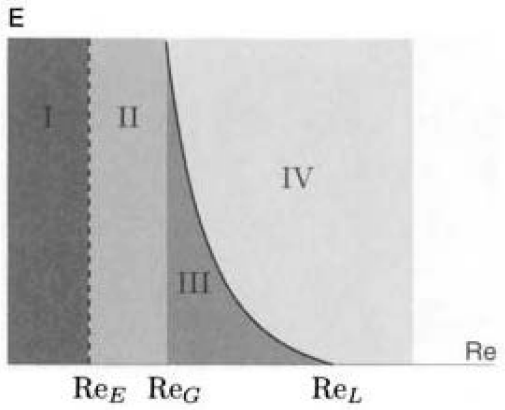

Figure 1: Critical Reynolds numbers for transitional flows

The Navier-Stokes equations are complete enough to describe the amplification of disturbances present in the original flow. In this example, LST will be applied to a three dimensional incompressible flow and then simplified.

Assumptions for the undisturbed flow and for the small disturbances:

1) Incompressible flow (ρ = constant)
2) Constant fluid properties/isothermal flow (μ,ν = constant)
3) Negligible body forces on the fluid
4) Laminar undisturbed flow
5) Infinitesimal disturbance velocities
6) Locally parallel flow around the disturbances

The governing equations for LST are the small disturbance equations. The small disturbance equations are Navier-Stokes equations modified by small one dimensional, sinusoidal wave disturbances, as shown in the derivation below.

Conservation of mass of the flow:

$$\nabla \cdot \boldsymbol{U_0} = 0 \qquad \textbf{[1]}$$

Conservation of momentum of the flow:

$$\frac{D\boldsymbol{U_0}}{dt} = -\frac{1}{\rho}\nabla p_0 + \nu\nabla^2\boldsymbol{U_0} \qquad \textbf{[2]}$$

The variables for velocity and pressure, in 3D:

$$\boldsymbol{U_0} = (U(t), V(t), W(t)), \qquad p_0 = p_x(t)$$

The small-disturbance velocity and pressure components:

$$\boldsymbol{u} = (\hat{u}(t), \hat{v}(t), \hat{w}(t)), \qquad \hat{p} = \hat{p}_x(t)$$

By substituting the small disturbance velocity and pressure into the Navier-Stokes equations and eliminating the terms identical to the undisturbed flow equations the continuity equation [3] and Navier-Stokes equations [4] including the small disturbance variables may be written.

$$\nabla \cdot (\boldsymbol{U_0} + \boldsymbol{u}) = \nabla \cdot \boldsymbol{U_0} \qquad \textbf{[3]}$$

$$\frac{D(\boldsymbol{U_0}+\boldsymbol{u})}{dt} + \frac{1}{\rho}\nabla(p_0 + \hat{p}) - \nu\nabla^2(\boldsymbol{U_0} + \boldsymbol{u}) = \frac{D\boldsymbol{U_0}}{dt} + \frac{1}{\rho}\nabla p_0 - \nu\nabla^2\boldsymbol{U_0} \qquad \textbf{[4]}$$

The continuity equation [3] reduces to equation [5] as follows:

$$\frac{\partial U}{\partial x} + \frac{\partial \hat{u}}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial \hat{v}}{\partial y} + \frac{\partial W}{\partial z} + \frac{\partial \hat{w}}{\partial z} = \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}$$

$$\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y} + \frac{\partial \hat{w}}{\partial z} = 0 \qquad \textbf{[5]}$$

The stream-wise (x) momentum equation is derived from equation [4] as shown below:

$$\frac{\partial U}{\partial t} + \frac{\partial \hat{u}}{\partial t} + U\frac{\partial U}{\partial x} + \hat{u}\frac{\partial U}{\partial x} + U\frac{\partial \hat{u}}{\partial x} + \hat{u}\frac{\partial \hat{u}}{\partial x} + V\frac{\partial U}{\partial y} + \hat{v}\frac{\partial U}{\partial y} + V\frac{\partial \hat{u}}{\partial y} + \hat{v}\frac{\partial \hat{u}}{\partial y} + \cdots$$

$$\ldots W\frac{\partial U}{\partial z} + \hat{w}\frac{\partial U}{\partial z} + W\frac{\partial \hat{u}}{\partial z} + \hat{w}\frac{\partial \hat{u}}{\partial z} + \frac{1}{\rho}\frac{\partial p_0}{\partial x} + \frac{1}{\rho}\frac{\partial \hat{p}}{\partial x} - \nu\nabla^2 U - \cdots$$

$$\ldots \nu\nabla^2 \hat{u} = \frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} + W\frac{\partial U}{\partial z} + \frac{1}{\rho}\frac{\partial p_0}{\partial x} - \nu\nabla^2 U$$

The equation reduces after subtraction of identical terms as shown below:

$$\frac{\partial \hat{u}}{\partial t} + \hat{u}\frac{\partial U}{\partial x} + U\frac{\partial \hat{u}}{\partial x} + \hat{u}\frac{\partial \hat{u}}{\partial x} + \hat{v}\frac{\partial U}{\partial y} + V\frac{\partial \hat{u}}{\partial y} + \hat{v}\frac{\partial \hat{u}}{\partial y} + \hat{w}\frac{\partial U}{\partial z} + W\frac{\partial \hat{u}}{\partial z} + \hat{w}\frac{\partial \hat{u}}{\partial z} + \cdots$$

$$\ldots \frac{1}{\rho}\frac{\partial \hat{p}}{\partial x} - \nu\nabla^2 \hat{u} = 0$$

The cross-stream (y) and span-wise (z) momentum equations are reduced by the same procedure.

Since the disturbances are assumed to be very small, all second order products are assumed to be negligible, linearizing the disturbance momentum equations.[3]The resulting momentum equations in Cartesian coordinates are linear partial differential equations, shown below in equations [6], [7], and [8].

$$\frac{\partial \hat{u}}{\partial t} + \hat{u}\frac{\partial U}{\partial x} + U\frac{\partial \hat{u}}{\partial x} + \hat{v}\frac{\partial U}{\partial y} + V\frac{\partial \hat{u}}{\partial y} + \hat{w}\frac{\partial U}{\partial z} + W\frac{\partial \hat{u}}{\partial z} + \frac{1}{\rho}\frac{\partial \hat{p}}{\partial x} - \nu\nabla^2 \hat{u} = 0 \qquad \textbf{[6]}$$

$$\frac{\partial \hat{v}}{\partial t} + \hat{u}\frac{\partial V}{\partial x} + U\frac{\partial \hat{v}}{\partial x} + \hat{v}\frac{\partial V}{\partial y} + V\frac{\partial \hat{v}}{\partial y} + \hat{w}\frac{\partial V}{\partial z} + W\frac{\partial \hat{v}}{\partial z} + \frac{1}{\rho}\frac{\partial \hat{p}}{\partial y} - \nu\nabla^2 \hat{v} = 0 \qquad \textbf{[7]}$$

$$\frac{\partial \hat{w}}{\partial t} + \hat{u}\frac{\partial W}{\partial x} + U\frac{\partial \hat{w}}{\partial x} + \hat{v}\frac{\partial W}{\partial y} + V\frac{\partial \hat{w}}{\partial y} + \hat{w}\frac{\partial W}{\partial z} + W\frac{\partial \hat{w}}{\partial z} + \frac{1}{\rho}\frac{\partial \hat{p}}{\partial z} - \nu\nabla^2 \hat{w} = 0 \qquad \textbf{[8]}$$

Since fully developed channel flow between infinite planes is the flow of interest it is safe to assume that the cross-stream velocity from shear layer to shear layer is (at least initially) negligible and that the stream-wise and span-wise velocities vary only in the cross-stream direction.[3]

$$U \approx U(y)$$

$$V \approx 0,\ \frac{\partial V}{\partial x} = \frac{\partial V}{\partial y} = \frac{\partial V}{\partial z} = 0$$

$$W \approx W(y)$$

For simplicity, the parallel flow assumption is also applied to the disturbances. The assumptions of parallel flow and parallel flow disturbances eliminates 10 convective terms from equations [6], [7], and [8], which become equations [10], [11], and [12]. For completeness, the three dimensional disturbance equations are shown below including continuity.

$$\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y} + \frac{\partial \hat{w}}{\partial z} = 0 \qquad \textbf{[10]}$$

$$\frac{\partial \hat{u}}{\partial t} + U\frac{\partial \hat{u}}{\partial x} + \hat{v}\frac{\partial U}{\partial y} + W\frac{\partial \hat{u}}{\partial z} = -\frac{1}{\rho}\frac{\partial \hat{p}}{\partial x} + \nu\nabla^2\hat{u} \qquad \textbf{[11]}$$

$$\frac{\partial \hat{v}}{\partial t} + U\frac{\partial \hat{v}}{\partial x} + W\frac{\partial \hat{v}}{\partial z} = -\frac{1}{\rho}\frac{\partial \hat{p}}{\partial y} + \nu\nabla^2\hat{v} \qquad \textbf{[12]}$$

$$\frac{\partial \hat{w}}{\partial t} + U\frac{\partial \hat{w}}{\partial x} + \hat{v}\frac{\partial W}{\partial y} + W\frac{\partial \hat{w}}{\partial z} = -\frac{1}{\rho}\frac{\partial \hat{p}}{\partial z} + \nu\nabla^2\hat{w} \qquad \textbf{[13]}$$

Substituting sinusoidal disturbances into the disturbance terms allows the three-dimensional terms (including *W* and its derivatives) to be eliminated by further substitution, as shown in Part (B). Therefore, the stability of any parallel flow, such as Poiseulle plane flow, can therefore be analyzed using the effective two dimensional flow.[3]

Part (B)

Derive the Orr-Sommerfeld equation and write the boundary conditions for a plane Poiseulle flow.

Solution:

Assuming the disturbances are sinusoidal the three dimensional, disturbances are described by equations [14], [15], [16], and [17], shown below:

$$\hat{u} = u(y)e^{i\alpha(xcos\varphi+zsin\varphi-ct)} \qquad \textbf{[14]}$$

$$\hat{v} = v(y)e^{i\alpha(xcos\varphi+zsin\varphi-ct)} \qquad \textbf{[15]}$$

$$\hat{w} = w(y)e^{i\alpha(xcos\varphi+zsin\varphi-ct)} \qquad \textbf{[16]}$$

$$\hat{p} = p(y)e^{i\alpha(xcos\varphi+zsin\varphi-ct)} \qquad \textbf{[17]}$$

Here, α is the wave number, φ is the direction of wave propagation in stream-wise and span-wise directions and *c* is the complex wave speed.[3] Substitution of equations [14], [15], [16], and [17] into the small disturbance equations allows for the small disturbance equations to be rewritten in modal form, shown below.

By substituting $\hat{u}, \hat{v}, \hat{w}$, and $\hat{p}$ into equation [10] forms equation [10.a] and further dropping the $e^{i\alpha(xcos\varphi+zsin\varphi-ct)}$ terms [10.a] becomes equation [18].

$$e^{i\alpha(xcos\varphi+zsin\varphi-ct)}\left(i\alpha(ucos\varphi + wsin\varphi) + \frac{dv}{dy}\right) = 0 \qquad \textbf{[10.a]}$$

$$i\alpha(ucos\varphi + wsin\varphi) + \frac{dv}{dy} = 0 \qquad \textbf{[18]}$$

Substituting $\hat{u}, \hat{v}, \hat{w}$, and $\hat{p}$ into equations [11], [12], and [13] and following the same procedure yields:

$$-i\alpha uc + i\alpha uUcos\varphi + v\frac{dU}{dy} + i\alpha uWsin\varphi = -\frac{1}{\rho}i\alpha pcos\varphi + \nu\left(\frac{d^2u}{dy^2} - \alpha^2 u\right) \qquad \textbf{[19]}$$

$$-i\alpha vc + i\alpha vUcos\varphi + i\alpha vWsin\varphi = -\frac{1}{\rho}\frac{dp}{dy} + \nu\left(\frac{d^2v}{dy^2} - \alpha^2 v\right) \qquad \textbf{[20]}$$

$$-i\alpha wc + i\alpha wUcos\varphi + v\frac{dW}{dy} + i\alpha wWsin\varphi = -\frac{1}{\rho}i\alpha psin\varphi + \nu\left(\frac{d^2w}{dy^2} - \alpha^2 w\right) \qquad \textbf{[21]}$$

Equations [19], [20], and [21] are rearranged for simplicity into equations [22], [23], and [24], shown below.

$$i\alpha u(-c + Ucos\varphi + Wsin\varphi) + v\frac{dU}{dy} = -\frac{1}{\rho} i\alpha p cos\varphi + \nu\left(\frac{d^2u}{dy^2} - \alpha^2 u\right) \quad \textbf{[22]}$$

$$i\alpha v(-c + Ucos\varphi + Wsin\varphi) = -\frac{1}{\rho}\frac{dp}{dy} + \nu\left(\frac{d^2v}{dy^2} - \alpha^2 v\right) \quad \textbf{[23]}$$

$$i\alpha w(-c + Ucos\varphi + Wsin\varphi) + v\frac{dW}{dy} = -\frac{1}{\rho} i\alpha p sin\varphi + \nu\left(\frac{d^2w}{dy^2} - \alpha^2 w\right) \quad \textbf{[24]}$$

To further simplify the notation, the fluctuating flow and disturbance velocity terms in the stream-wise and span-wise directions are grouped, and the *0* subscript is used to denote the compact notation.[3]

$$u_0 = ucos\varphi + wsin\varphi \quad \textbf{[25]}$$

$$U_0 = Ucos\varphi + Wsin\varphi \quad \textbf{[26]}$$

Substitution of equations [25] and [26] into equations [18],[22], [23], and [24], the linear ordinary differential equation (ODE) form of the disturbance equations with complex coefficients are reduced as shown below.

$$i\alpha(u_0) + \frac{dv}{dy} = 0 \quad \textbf{[27]}$$

$$i\alpha u(-c + U_0) + v\frac{dU}{dy} = -\frac{1}{\rho} i\alpha p cos\varphi + \nu\left(\frac{d^2u}{dy^2} - \alpha^2 u\right) \quad \textbf{[28]}$$

$$i\alpha v(-c + U_0) = -\frac{1}{\rho}\frac{dp}{dy} + \nu\left(\frac{d^2v}{dy^2} - \alpha^2 v\right) \quad \textbf{[29]}$$

$$i\alpha w(-c + U_0) + v\frac{dW}{dy} = -\frac{1}{\rho} i\alpha p sin\varphi + \nu\left(\frac{d^2w}{dy^2} - \alpha^2 w\right) \quad \textbf{[30]}$$

As noted by White and mentioned in the solution to Part (A), the disturbance equations possess a unique property. By multiplying equation [28] by cos(φ) and equation [30] by sin(φ)a series of operations may be performed to reduce the four disturbance equations into a single equation.

$$i\alpha u(-c + U_0)cos\varphi + v\frac{dU}{dy}cos\varphi = -\frac{1}{\rho} i\alpha p cos^2\varphi + \nu\left(\frac{d^2u}{dy^2} - \alpha^2 u\right)cos\varphi \quad \textbf{[31]}$$

$$i\alpha w(-c + U_0)sin\varphi + v\frac{dW}{dy}sin\varphi = -\frac{1}{\rho} i\alpha p sin^2\varphi + \nu\left(\frac{d^2w}{dy^2} - \alpha^2 w\right)sin\varphi \quad \textbf{[32]}$$

Further simplification by adding equations [30] and [31]:

$$i\alpha u(-c + U_0)cos\varphi + v\frac{dU}{dy}cos\varphi + i\alpha w(-c + U_0)sin\varphi + v\frac{dW}{dy}sin\varphi =$$

$$= -\frac{1}{\rho} i\alpha p cos^2\varphi + \nu\left(\frac{d^2u}{dy^2} - \alpha^2 u\right) cos\varphi + -\frac{1}{\rho} i\alpha p sin^2\varphi + \nu\left(\frac{d^2w}{dy^2} - \alpha^2 w\right) sin\varphi$$

Which reduces further by once again using the compact notation of equations [25] and [26], shown below.

$$i\alpha u_0(-c + U_0) + \nu\frac{dU_0}{dy} = -\frac{1}{\rho} i\alpha p + \nu\left(\frac{d^2u_0}{dy^2} - \alpha^2 u_0\right) \qquad \textbf{[33]}$$

The disturbance equations may then be rewritten as functions of u_0, v, and p, confirming that the stability of a three dimensional parallel flow can be found by two dimensional analysis.[4]For completeness, the simplified disturbance equations [27], [29], and [33] are shown together below.

$$i\alpha(u_0) + \frac{dv}{dy} = 0 \qquad \textbf{[34]}$$

$$i\alpha u_0(-c + U_0) + v\frac{dU_0}{dy} = -\frac{1}{\rho} i\alpha p + \nu\left(\frac{d^2u_0}{dy^2} - \alpha^2 u_0\right) \qquad \textbf{[35]}$$

$$i\alpha v(-c + U_0) = -\frac{1}{\rho}\frac{dp}{dy} + \nu\left(\frac{d^2v}{dy^2} - \alpha^2 v\right) \qquad \textbf{[36]}$$

The Orr-Sommerfeld relation is formed by combining the three equations into a single 4th order, homogenous, linear differential equation which is a function of the cross-stream velocity alone. To form the equation, u_0 and p must be eliminated by substitution. First, equation [34] is rearranged first to solve for u_0:

$$u_0 = -\frac{1}{i\alpha}\frac{dv}{dy} \qquad \textbf{[37]}$$

Equation [37] is then substituted into equation [35] and simplified by multiplying the equation by $i\alpha$, as shown below.

$$-\frac{dv}{dy}(-c + U_0) + v\frac{dU_0}{dy} = -\frac{1}{\rho} i\alpha p + \nu\left(-\frac{d^3v}{dy^3}\frac{1}{i\alpha} + \alpha^2\frac{1}{i\alpha}\frac{dv}{dy}\right)$$

$$-i\alpha\frac{dv}{dy}(U_0 - c) + i\alpha v\frac{dU_0}{dy} = \frac{1}{\rho}\alpha^2 p + \nu\left(\alpha^2 - \frac{d^2v}{dy^2}\right)\frac{dv}{dy} \qquad \textbf{[38]}$$

In order to eliminate p in the equation [38] by substitution, the derivative with respect to y must be taken to form equation [39].

$$-i\alpha\frac{d}{dy}\left(\frac{dv}{dy}(U_0 - c)\right) + \frac{d}{dy}\left(i\alpha v\frac{dU_0}{dy}\right) = \frac{\alpha^2}{\rho}\frac{dp}{dy} + \nu\left(\alpha^2 - \frac{d^2v}{dy^2}\right)\frac{d^2v}{dy^2}$$

$$-i\alpha\frac{d^2v}{dy^2}(U_0-c)-i\alpha\frac{dv}{dy}\frac{dU_0}{dy}+i\alpha\frac{dv}{dy}\frac{dU_0}{dy}+i\alpha v\frac{d^2U_0}{dy^2}=\frac{\alpha^2}{\rho}\frac{dp}{dy}+\nu\left(\alpha^2-\frac{d^2v}{dy^2}\right)\frac{d^2v}{dy^2}$$

$$-i\alpha\tfrac{d^2v}{dy^2}(U_0-c)+i\alpha v\tfrac{d^2U_0}{dy^2}=\tfrac{\alpha^2}{\rho}\tfrac{dp}{dy}+\nu\left(\alpha^2-\tfrac{d^2v}{dy^2}\right)\tfrac{d^2v}{dy^2}\qquad\textbf{[39]}$$

Equation [36] is reordered to equate to $\frac{dp}{dy}$.

$$\tfrac{dp}{dy}=-\rho i\alpha v(U_0-c)+\rho\left(\tfrac{d^2v}{dy^2}-\alpha^2v\right)\qquad\textbf{[40]}$$

p is then eliminated by the substitution of equation [40] into equation [39].

$$-i\alpha\frac{d^2v}{dy^2}(U_0-c)+i\alpha v\frac{d^2U_0}{dy^2}=\frac{\alpha^2}{\rho}\left(-\rho i\alpha v(U_0-c)+\rho\nu\left(\frac{d^2v}{dy^2}-\alpha^2v\right)\right)\ldots$$
$$\ldots+\nu\left(\alpha^2-\frac{d^2v}{dy^2}\right)\frac{d^2v}{dy^2}$$

$$-i\alpha\frac{d^2v}{dy^2}(U_0-c)+i\alpha v\frac{d^2U_0}{dy^2}$$
$$=-i\alpha^3v(U_0-c)+\nu\alpha^2\left(\frac{d^2v}{dy^2}-\alpha^2v\right)+\nu\left(\alpha^2-\frac{d^2v}{dy^2}\right)\frac{d^2v}{dy^2}$$

Shifting all terms to one side:

$$\left(i\alpha^3v-i\alpha\frac{d^2v}{dy^2}\right)(U_0-c)+i\alpha v\frac{d^2U_0}{dy^2}-\nu\alpha^2\left(\frac{d^2v}{dy^2}-\alpha^2v\right)-\nu\left(\alpha^2-\frac{d^2v}{dy^2}\right)\frac{d^2v}{dy^2}=0$$

Dividing by $-\frac{1}{i\alpha}$:

$$\left(\frac{d^2v}{dy^2}-\alpha^2v\right)(U_0-c)-v\frac{d^2U_0}{dy^2}+\frac{\nu}{i}\left(\alpha\frac{d^2v}{dy^2}-\alpha^3v\right)+\frac{\nu}{i}\left(\alpha-\frac{1}{\alpha}\frac{d^2v}{dy^2}\right)\frac{d^2v}{dy^2}=0$$

Regrouping:

$$\left(\frac{d^2v}{dy^2}-\alpha^2v\right)(U_0-c)-v\frac{d^2U_0}{dy^2}+\frac{\nu}{i}\left(\alpha\frac{d^2v}{dy^2}-\alpha^3v+\alpha\frac{d^2v}{dy^2}-\frac{1}{\alpha}\frac{d^4v}{dy^4}\right)=0$$

Factoring:

$$\left(\frac{d^2v}{dy^2}-\alpha^2v\right)(U_0-c)-v\frac{d^2U_0}{dy^2}-\frac{\nu}{i\alpha}\left(\alpha^2\frac{d^2v}{dy^2}-\alpha^4v+\alpha^2\frac{d^2v}{dy^2}-\frac{d^4v}{dy^4}\right)=0$$

And rearranging to produce equation [41]:

$$\left(\frac{d^2v}{dy^2} - \alpha^2 v\right)(U_0 - c) - v\frac{d^2U_0}{dy^2} - \frac{\nu}{i\alpha}\left(\frac{d^4v}{dy^4} - 2\alpha^2\frac{d^2v}{dy^2} + \alpha^4 v\right) = 0 \qquad \textbf{[41]}$$

Equation [41] is the Orr-Sommerfeld Equation. The units of the equation above are $1/s^2$, however, the non-dimensional version of the equation is formed by dividing equation [41] by $\frac{h^2}{U_m^2}$, the squares of the channel height divided by the *mean* velocity in the stream-wise direction.

$$\Phi = \frac{v(y)}{U_m},\ \bar{y} = \frac{y}{h},\ \overline{U} = \frac{U_0}{U_m},\ \overline{c} = \frac{c}{U_m},\ \bar{\alpha} = \alpha h,\ Re = \frac{U_m h}{\nu}$$

$$\left(\frac{d^2\Phi}{d\bar{y}^2} - \bar{\alpha}^2\Phi\right)(\overline{U} - \bar{c}) - \Phi\frac{d^2\overline{U}}{d\bar{y}^2} - \frac{1}{i\bar{\alpha}Re}\left(\frac{d^4\Phi}{d\bar{y}^4} - 2\bar{\alpha}^2\frac{d^2\Phi}{d\bar{y}^2} + \bar{\alpha}^4\Phi\right) = 0$$

And rearranging the non-dimensional Orr-Somerfield Equation:

$$\frac{d^4\Phi}{d\bar{y}^4} - 2\bar{\alpha}^2\frac{d^2\Phi}{d\bar{y}^2} + \bar{\alpha}^4\Phi - i\bar{\alpha}Re\left(\left(\frac{d^2\Phi}{d\bar{y}^2} - \bar{\alpha}^2\Phi\right)(\overline{U} - \bar{c}) - \Phi\frac{d^2\overline{U}}{d\bar{y}^2}\right) = 0$$

Since all of the derivatives are in respect to the *y* direction, normal to the flow, the shorter derivative notation is implemented for simplicity:

$$\Phi'''' - 2\bar{\alpha}^2\Phi'' + \bar{\alpha}^4\Phi - i\bar{\alpha}Re\left((\Phi'' - \bar{\alpha}^2\Phi)(\overline{U} - \bar{c}) - \Phi\overline{U}''\right) = 0$$

Noting that the wave frequency ω is equal to the wavenumber $\bar{\alpha}$ multiplied by the wave velocity $\bar{c}$, the equation may be rewritten to match the arrangement used by Bradshaw.[5]

$$\overline{\omega} = \bar{\alpha}\bar{c}$$

$$\Phi'''' - 2\bar{\alpha}^2\Phi'' + \bar{\alpha}^4\Phi - iRe\left((\Phi'' - \bar{\alpha}^2\Phi)(\bar{\alpha}\overline{U} - \overline{\omega}) - \bar{\alpha}\Phi\overline{U}''\right) = 0 \quad \textbf{[42]}$$

For simplicity, the bars will be dropped from the non-dimensional terms for the remainder of this study:

$$\Phi'''' - 2\alpha^2\Phi'' + \alpha^4\Phi - iRe\left((\Phi'' - \alpha^2\Phi)(\alpha U - \omega) - \alpha\Phi U''\right) = 0$$

Part (C)
For the spatial amplification case in plane Poiseuille flow, compute and plot the neutral stability curve between Re = 5000 and 50,000, where Re = Uh/υ with U = average velocity and h = channel spacing. Use the numerical method of your choice, but carefully explain the solution procedure.

Solution:

To analyze the Orr-Sommerfeld equation the complex components of the wave number, the complex components of the wave frequency, and the Reynolds number (five scalars) are iteratively guessed to generate non-trivial solutions.[1] For the case of spatial amplification, the temporal amplification is neglected ($\omega_i = 0$). Although the largest rate of spatial amplification occurs when $\omega_i = 0$ and $\alpha_i >> 1$, the neutral stability curve is of interest because it is the locus which separates the damped-disturbance region from the amplified-disturbance region.[1] The spatial amplification neutral curve corresponds to values ω_r, α_r , and Re and the complex wave number α_i is zero, and describes a flow in which the disturbance will propagate through parallel mean flow with constant amplitude.

To generate the neutral stability curve, the following steps were taken:

1) Determination of appropriate boundary and initial conditions for the Orr-Sommerfeld equation.
2) Discretization of the Orr-Sommerfeld equation into four first order ODEs in a central difference format.
3) Generation of an appropriate computational grid, capable of accurately modeling the critical layer in the flow were disturbances grow rapidly.[6]
4) Computation of the eigenvalues of the four first order ODEs using Keller's Box Method.
5) Iterating by Newton's method to find non-trivial solutions for ω_r and α_r for a given Reynolds number.[1]

For the first step, the Poiseuille plane flow non-dimensional streamwise velocity profile is normalized about the mean to provide an initial flow condition, shown below.

$$U(y) = 1 - 6\left(y - \tfrac{1}{2}\right)^2 \text{ for } y = \{0, 0.001, 0.002, \ldots, 1\}$$

$$U_{max}(y = 1/2\,) = 1.5$$

$$U''(y) = -12$$

$$U_{mean}(y) = \frac{1}{1-0}\int_0^1 1 - 6\left(y - \frac{1}{2}\right)^2 dy = \int_0^1 -6y^2 + 6y\, dy = -2 + 3 = 1$$

The boundary conditions for the case of plane Poiseuille flow for the top of the channel are specified as no-slip and no-penetration conditions. However, the no-penetration

condition is relaxed and substituted by another condition to allow for the computation of non-trivial solutions.

$\varphi = 0,$ at $y = h$

$\varphi'' - \alpha^2\varphi = 1,$ at $y = h$

At the bottom plane, no slip and no penetration are enforced:

$\varphi = 0,$ at $y = 0$

$\varphi' = 0,$ at $y = 0$

Second, the Orr-Sommerfeld equation is split into four single order ordinary differential equations, shown below in matrix form. The unknown variables to be found are written on the left hand side (LHS).

$$\begin{bmatrix} \varphi \\ s \\ f \\ g \end{bmatrix} = \begin{bmatrix} \varphi \\ f' - \alpha^2\varphi \\ \varphi' \\ s' \end{bmatrix}$$

Which can be rearranged in terms of the first derivative of each unknown.

$$\begin{bmatrix} \varphi' \\ s' \\ f' \\ g' \end{bmatrix} = \begin{bmatrix} f \\ g \\ s + \alpha^2\varphi \\ \zeta^2 s - iRe\alpha U''\varphi \end{bmatrix} \qquad \textbf{[43]}$$

$$\zeta^2 = \alpha^2 + iRe(\alpha U - \omega) \qquad \textbf{[44]}$$

Equations [43] and [44] are formed in such a way that by substitution of variables the equations may be rearranged back into the Orr-Sommerfeld equation [42].

The difference equations describing the change from point to point, with 2nd order accuracy, are then written as shown below for the top row of equation [43].

$\varphi' = f$

$$\frac{\varphi_{i+1} - \varphi_i}{\bar{y}_{i+1} - \bar{y}_i} = \frac{f_{i+1} + f_i}{2}$$

Equation [45] is the rewritten in order of ascending location indices of unknown variables, bringing all terms to the left hand side of the equation.

$$-2\varphi_i - (\bar{y}_{i+1} - \bar{y}_i)f_i + 2\varphi_{i+1} - (\bar{y}_{i+1} - \bar{y}_i)f_{j+1} = 0 \qquad \textbf{[45]}$$

The second row of equation [43] is then also approximated as equation [46].

$$s' = g$$

$$\frac{s_{i+1} - s_i}{\bar{y}_{i+1} - \bar{y}_i} - \frac{g_{i+1} + g_i}{2} = 0$$

$$-2s_i - (\bar{y}_{i+1} - \bar{y}_i)g_i + 2s_{i+1} - (\bar{y}_{i+1} - \bar{y}_i)g_{i+1} = 0 \quad \textbf{[46]}$$

By the same process, the third row of [43] is then also approximated as equation [47].

$$f' = s + \alpha^2\varphi$$

$$\frac{f_{i+1} - f_i}{\bar{y}_{i+1} - \bar{y}_i} - \frac{s_{i+1} + s_i}{2} - \alpha^2\left(\frac{\varphi_{i+1} + \varphi_i}{2}\right) = 0$$

$$-\alpha^2(\bar{y}_{i+1} - \bar{y}_i)\varphi_i - 2f_i - (\bar{y}_{i+1} - \bar{y}_i)s_i - \alpha^2(\bar{y}_{i+1} - \bar{y}_i)\varphi_{i+1} + 2f_{i+1} \dots$$

$$\dots - (\bar{y}_{i+1} - \bar{y}_i)s_{i+1} = 0 \quad \textbf{[47]}$$

The fourth row of [43] is then also approximated as shown below.

$$g' = \zeta^2 s - iReU''\varphi$$

$$\frac{g_{i+1} - g_i}{\bar{y}_{i+1} - \bar{y}_i} - \left(\alpha^2 + iRe\left(\alpha\left(\frac{U_{i+1} + U_i}{2}\right) - \omega\right)\right) \cdot \left(\frac{s_{i+1} + s_i}{2}\right) + \cdots$$

$$\dots + iRe\alpha U''\left(\frac{\varphi_{i+1} + \varphi_i}{2}\right) = 0$$

By rearranging and distributing terms, equation [48] is formed.

$$iRe\alpha U''\varphi_i - (\bar{y}_{i+1} - \bar{y}_i)\left(\alpha^2 + iRe\left(\alpha\left(\frac{U_{i+1} + U_i}{2}\right) - \omega\right)\right)s_i - 2g_i + \cdots$$
$$\dots + iRe\alpha U''\varphi_{i+1} - (\bar{y}_{i+1} - \bar{y}_i)\left(\alpha^2 + iRe\left(\alpha\left(\frac{U_{i+1}+U_i}{2}\right) - \omega\right)\right)s_{i+1} + 2g_{i+1} = 0 \textbf{[48]}$$

Equations [45] through [48] are the central difference equations for the Orr-Sommerfeld, which are then distributed throughout the triadiagonal "box" described by Keller,[5] filling in the entire computational domain except for two rows of boundary conditions at the top and bottom of the grid.

For the third step, a Gauss-Lobatto grid was automatically generated using the following equation, were {1:N} represents each gridpoint. Figures 2 and 3 show the grid beginning

at $i = 1$, for the bottom portion of a Poiseulle channel flow. Also shown below is grid generation formula for a set of gridpoints {1,2,3,...,N}.

$$y_{Gridpoint}(\{1{:}\,N\}) = \frac{1}{2} + \frac{\cos\left(\frac{\{0:N-1\}}{N-1}\right)}{2}$$

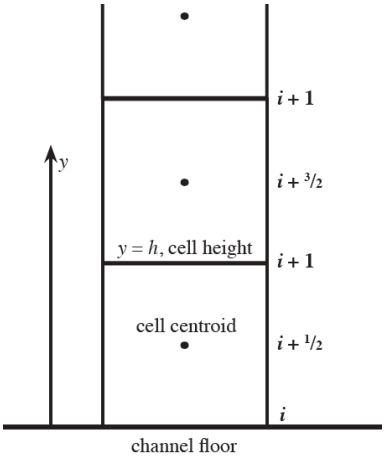

Figure 2: Computational grid (not scaled)

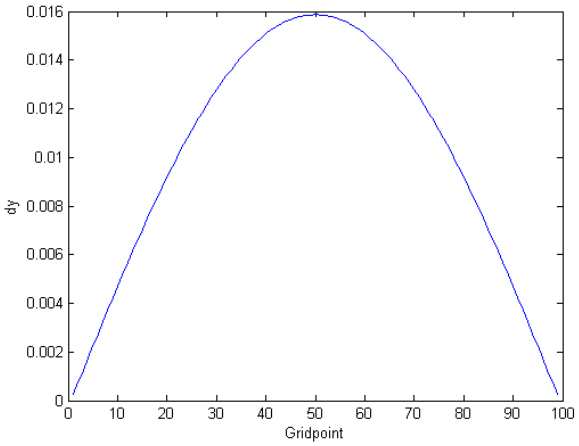

Figure 3: Change in y, gridpoint height, across the channel for $N = 100$.

The difference equations are staggered in a matrix of matrices, referred to as the equation matrix, such that the first two rows of equation [43] compose the last two rows of the first sub-matrix of the equation matrix.[1] The first two rows of the first matrix are the channel top boundary conditions. The last two rows of equation [43] correspond to the first two rows of the next sub-matrix, and so on down to the boundary conditions at the channel floor. A miniature, [3x3] version of the tridiagonal equation matrix for computation is shown in the Appendix I. The complete central difference equations are shown again below in the order in which they appear in the equation matrix and the numerical code.

$$-2\varphi_i - (\bar{y}_{i+1} - \bar{y}_i)f_i + 2\varphi_{i+1} - (\bar{y}_{i+1} - \bar{y}_i)f_{j+1} = 0$$

$$-\alpha^2(\bar{y}_{i+1}-\bar{y}_i)\varphi_i - 2f_i - (\bar{y}_{i+1}-\bar{y}_i)s_i - \alpha^2(\bar{y}_{i+1}-\bar{y}_i)\varphi_{i+1} + 2f_{i+1} - (\bar{y}_{i+1}-\bar{y}_i)s_{i+1} = 0$$

$$-2s_i - (\bar{y}_{i+1}-\bar{y}_i)g_i + 2s_{i+1} - (\bar{y}_{i+1}-\bar{y}_i)g_{i+1} = 0$$

$$iRe\alpha U''\varphi_i - (\bar{y}_{i+1}-\bar{y}_i)\left(\alpha^2 + iRe\left(\alpha\left(\frac{U_{i+1}+U_i}{2}\right)-\omega\right)\right)s_i - 2g_i + \cdots$$

$$\ldots + iRe\alpha U''\varphi_{i+1} - (\bar{y}_{i+1}-\bar{y}_i)\left(\alpha^2 + iRe\left(\alpha\left(\frac{U_{i+1}+U_i}{2}\right)-\omega\right)\right)s_{i+1} + 2g_{i+1} = 0$$

The difference equations may be solved for unknown eigenvalues when written in matrix form, shown below, where ***A*** is the equation matrix shown in the Appendix of *N*x*N* dimensions.

$$\boldsymbol{A}\cdot\Delta = \boldsymbol{r} \quad \textbf{[49]}$$

The fourth and fifth steps were completed by a computer using custom made multi-matrix LU factorization functions, summation functions, and multiplication functions in the numerical code.[1] Keller's Box Method uses Lower and Upper tridiagonal (LU) factorization to solve equation [49] for an unknown vector of eigenvectors, δ, as shown below.[1]

$$\boldsymbol{A}\cdot\delta = \boldsymbol{L}\cdot\boldsymbol{U}\cdot\delta = \boldsymbol{r}$$

Where:

$$\boldsymbol{U}\cdot\delta = \boldsymbol{w} \quad \textbf{[50]}$$

$$\boldsymbol{L}\cdot\boldsymbol{w} = \boldsymbol{r} \quad \textbf{[51]}$$

Equations [50] and [51] are more conveniently solved than equation [49] because the *L* and *U* matrices each are composed of only two diagonal sets of sub-matrices. The ***A*** matrix contains diagonals *B,A,C*, going from left to right. For a grid of *N* gridpoints (i,i+1,i+2,...I-1, I), there will be *N* matrices of diagonal *A* sub-matrices and *N-1* matrices in diagonals *B* and *C* within the equation matrix. The ***L*** matrix is composed of the diagonals Δ and *C*, where Δ runs down the center diagonal and is *N* matrices in length. The *U* matrix is composed of *T* and *I* matrices, where *T* runs down the one-below center diagonal and the set of identity matrices *I* run down the center diagonal. The LU factorization is then formed by the numerical code following the equations below, where subscripts indicate the location of the sub-matrix from the top down.

Sweeping down the equation matrix:

$$\Delta_0 = \boldsymbol{A}_0$$

$$T_i = B_i/\Delta_{i-1}$$

$$\Delta_i = \boldsymbol{A}_i - T_i \cdot C_{i-1}$$

To solve equations [50], a downwards sweep is made for the $\boldsymbol{L}$ matrix row of T matrices:

$$\boldsymbol{w}_0 = \boldsymbol{r}_0$$

$$\boldsymbol{w}_i = \boldsymbol{r}_i - T_i \cdot \boldsymbol{w}_{i-1}$$

Followed by an upwards sweep of the Δ and C diagonals of the $\boldsymbol{U}$ matrix to solve for the unknown eigenvalues of δ.

$$\delta_I = \Delta_I \backslash \boldsymbol{w}_I$$

$$\delta_i = \Delta_i \backslash \boldsymbol{w}_i - \Delta_i \backslash (C_i \cdot \delta_{i-1}) \quad \textbf{[52]}$$

As $s = 1$, was the boundary condition set at the top of the channel, the no-penetration boundary condition (f_0=0) was not enforced. Therefore, the values of α_r and ω_r are iterated for a given Reynolds number such that complex value of f_0 is driven to zero using Newton's method, shown below by increments j of $\delta\alpha$ and $\delta\omega$.[5]

$$f_0 = f_r + if_i$$

$$\delta{\alpha_r}^j = \left(\frac{\left(f_i \frac{\partial f_r}{\partial \omega_r} - f_r \frac{\partial f_i}{\partial \omega_r} \right)}{\frac{\partial f_r}{\partial \alpha_r}\frac{\partial f_i}{\partial \omega_r} - \frac{\partial f_i}{\partial \alpha_r}\frac{\partial f_r}{\partial \omega_r}} \right)^j$$

$$\delta{\omega_r}^j = \left(\frac{\left(f_r \frac{\partial f_i}{\partial \alpha_r} - f_i \frac{\partial f_r}{\partial \alpha_r} \right)}{\frac{\partial f_r}{\partial \alpha_r}\frac{\partial f_i}{\partial \omega_r} - \frac{\partial f_i}{\partial \alpha_r}\frac{\partial f_r}{\partial \omega_r}} \right)^j$$

Here, $\delta\alpha_r$ and $\delta\omega_r$ are extracted from the derivatives of f_r and f_i at the same physical locations in the new unknown vectors shown below, which again require LU factorization of the equation matrix to solve for a new set of unknowns.

$$\boldsymbol{A} \cdot \frac{\partial \delta}{\partial \alpha_r}^j = -\frac{\partial \boldsymbol{A}}{\partial \alpha_r}^j \cdot \delta^j = \boldsymbol{r}_\alpha^j \quad \textbf{[54]}$$

$$\boldsymbol{A} \cdot \frac{\partial \delta}{\partial \omega_r}^j = -\frac{\partial \boldsymbol{A}}{\partial \omega_r}^j \cdot \delta^j = \boldsymbol{r}_\omega^j \quad \textbf{[55]}$$

All codes, carefully annotated and following the method above, are attached in the Appendix II. A desktop computer (Windows 7, 64-bit, Intel Core i7-2600, 3.40Gz, 8GB RAM) was used to generate results. The code flowchart is shown in Figure 3.

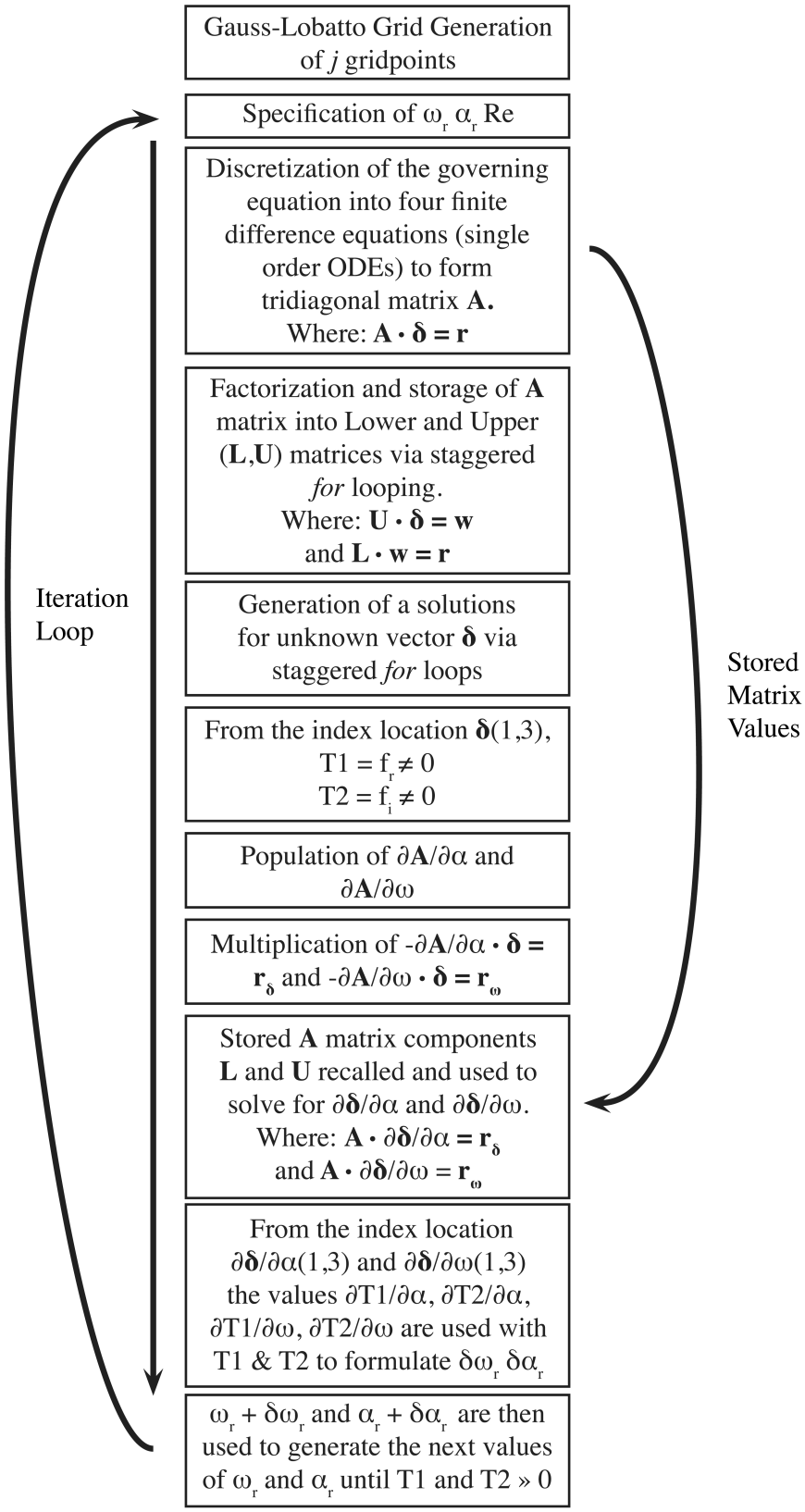

Figure 4: Keller's box method flow chart

Part (D)
Identify the critical Reynolds number in your computations, and discuss the discrepancy between the predicted and measured values, including ideas and results from recent literature.

Solution:

As mentioned in Part (A), the critical Reynolds number determined by LST is a function of a both the wave number and wave frequency of the instabilities. Small, random disturbances are modeled as infinitesimal instabilities. For some wave number and wave frequency combinations, the instabilities will be damped while other wave number and wave frequency combinations cause be the instabilities to become amplified, possibly resulting in transition.[7] It is important to note that LST predicts the demise of laminar flow only, not the onset of turbulence.[3]

The minimum flow condition for amplification of small disturbances occurs when the action of viscosity creates a critical layer in the flow (where the fluid velocity in the channel equals the real component of the complex phase velocity) in which one at least eigenmode of the complex phase velocity is sustained.[6] Although the existence of a critical layer has yet to be confirmed by experiment,[6] the existence of the aforementioned eigenmode, also known as Tollmien-Schlichting waves, has been validated by experiment.[2] Tollmien-Schlichting waves occur in conditions corresponding to the neutral stability curve described in Part (C). Table I shows approximate agreement between referenced values for the critical Reynolds number of plane Poiseuille flow, based on full channel height for the characteristic length and the mean velocity for the characteristic velocity.

Table 1: Adjusted reference values for the critical Reynolds number

Re_{crit}	$\alpha_{r\,crit}$	$\omega_{r\,crit}$ *	Method	Source	Citation
~8000	~2.04	N/A	Numerical	Cebeci and Bradshaw	[1]
7696.2	2.040	0.8067	Numerical	Schmid and Henningson	[2]
~8000	~2.04	N/A	Numerical	Godfrey, Samuels, and Barenghi	[8]

* c_r, the real component of the complex phase velocity was provided on a neutral stability curve.

Shown below are "thumb" curves, showing the critical Reynolds numbers at the lowest Reynolds numbers on the curve. Note that the curves shown below are all based on full channel height for the characteristic length and the mean velocity for the characteristic velocity.

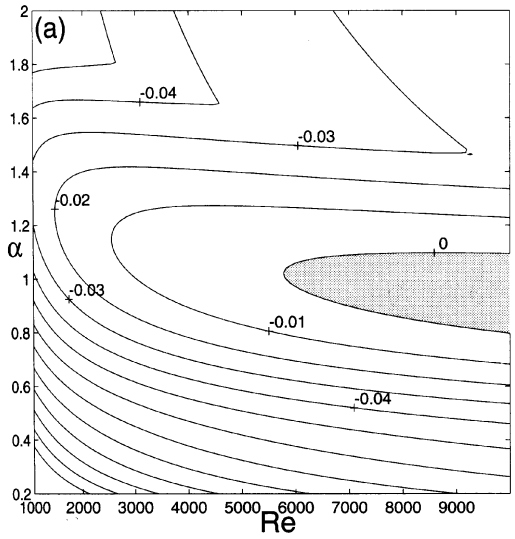

Figure 5: Contours of constant growth rate c_i (shaded area unstable) [2]

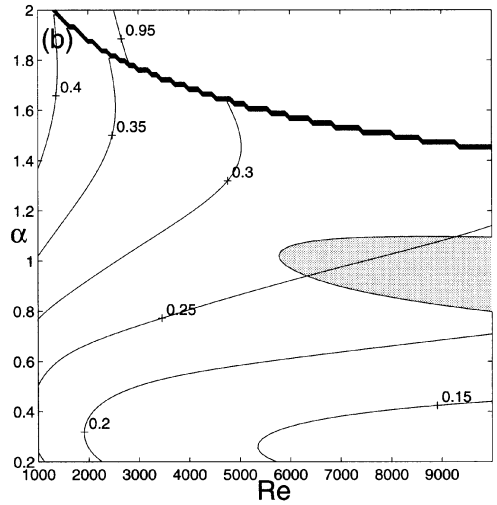

Figure 6: Contours of constant phase velocity c_r (shaded area unstable) [2]

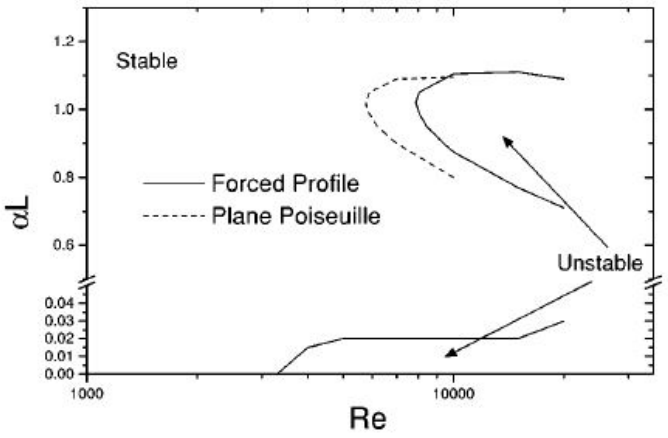

Figure 7: Neutral stability curve ($c_i = 0$) for a forced normal fluid profile and a plane Poiseuille flow for comparision [8]

The results of the numerical method described in Part (C) are in agreement with referenced Reynolds number, wave number, and wave frequencies. The method is at minimum second order accurate, as shown by error calculations in Table 2. Also shown below is the computed neutral stability curve, plotted against wave number and wave frequency.

Table 2: Critical value results

Re_{crit}	$\alpha_{r\ crit}$	$\omega_{r\ crit}$	N, gridpoints	Error (Schmid (2001)
7247.5	2.063	0.8310	100	5.83%, 1.12%, 3.01%
7581.0	2.044	0.8130	200	1.50%, 0.20%, 0.08%
7668.0	2.038	0.8071	300	0.36%, 0.10%, 0.05%

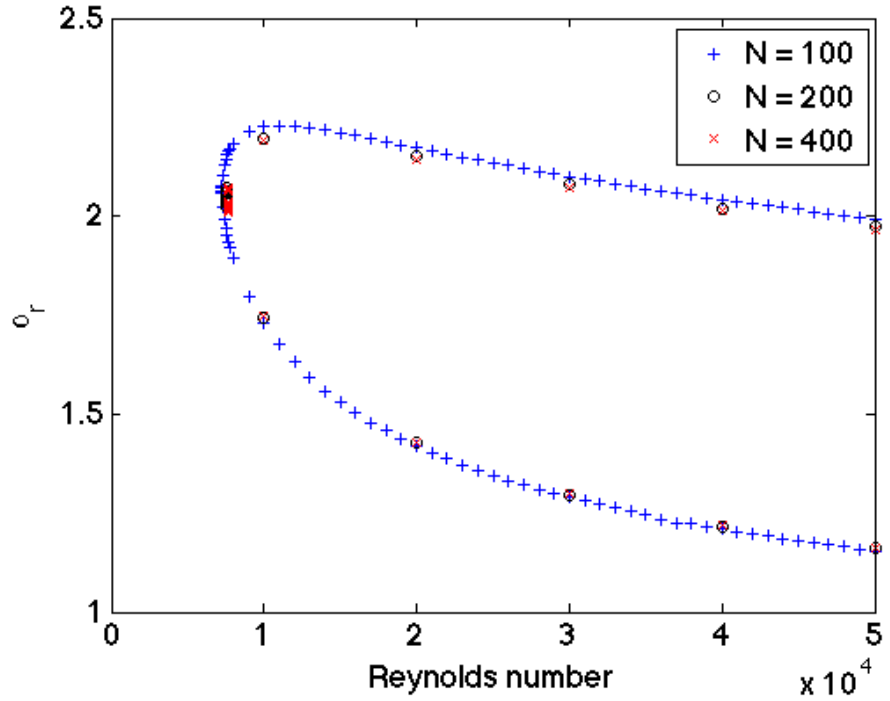

Figure 8: Neutral stability curve result plot for wave number

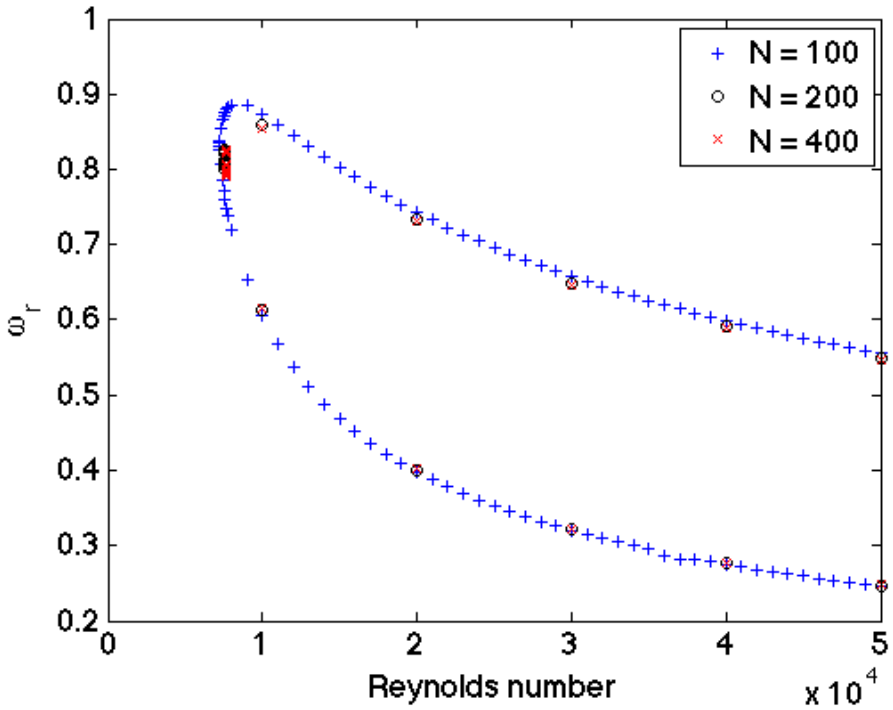

Figure 9: Neutral stability curve result plot for wave frequency

Numerical Problems and Troubleshooting

As noted by White, competing small and large solutions to the viscous Orr-Sommerfeld equation can be challenging.[3] The selected programming language, Mathworks MATLAB R2012a, defaults to double precision accuracy (eight bytes) which provides roughly 15-16 digits of precision. Checkpoints were included in the code to verify equations [49], [54], and [55]. However, the numerical codes did not provide reasonable answers for due to either a problem with validation (errors in the set up of the physics of the problem) or a problem with verification (errors in the numerical execution of the problem).[9] Possible validation errors are listed below:

1. Errors in the physical model,[9] resulting from the neglection of non-linear terms in the small-disturbance equations of LST and other physical assumptions listed in Part (A). By definition, LST results will differ at least slightly from experiments.

Possible verification problems are listed below:

1. Discretization errors resulting from coarse grids.[6] As grids of $N = 100$ were tested, further results from fine grids will reveal whether or not the critical layer grid resolution is appropriate.

2. Round-off errors resulting from limited available computational memory.[9] Round-off errors were not a source of error, as equation [49] was verified by multiplicative identity to 10^{-13} and equations [54] and [55] were verified by multiplicative identities to 10^{-15} using the double precision, the MATLAB R2012a default precision setting.

References

[1] Cebeci T. and Bradshaw P., *Momentum Transfer in Boundary Layers*, Hemisphere Publishing Corporation, 1977.

[2] Schmid, P., and Henningson, D., *Stability and Transition in Shear Flows*, Applied Mathematical Sciences, Vol. 142, Springer-Verlag Inc., 2001.

[3] White, F., *Viscous Fluid Flow*, McGraw-Hill, 3rd Edition, 2006.

[4] Gregory, N., Stuart, J.T., and Walker, W.S., *On the Stability of Three-Dimensional Boundary Layers with Applications to the Flow Due to a Rotating Disc*, Philosophical Transaction of the Royal Society of London A, Vol. 248, pp 155-199.

[5] Bradshaw, P., Cebeci, T., and Whitelaw, J., *Engineering Calculation Methods for Turbulent Flow*, Academic Press Inc., 1981.

[6] Muralidhar, K., and Biswas, G., *Advanced Engineering Fluid Mechanics*, Alpha Science International, Ltd., 2nd Edition, 2005.

[7] Schetz, J., and Bowersox, R., *Boundary Layer Analysis*, American Institute of Aeronautics and Astronautics, 2nd Edition, 2011.

[8] Godfrey, S., Samuels, D., and Barenghi, C., *Linear Stability of Laminar Plane Poiseuille Flow of Helium II under Non-Uniform Mutual Friction Forcing*, Physics of Fluids, Vol. 13.4, 2001.

[9] Zikanov, O. *Essential Computational Fluid Dynamics*, John Wiley & Sons, Inc., 2010.

Appendix I

Equation Matrices

Appendix II

Numerical Codes, Mathworks MATLAB R012a

1) Poiseulle Stability, primary script

```
% Numerical Analysis of Stability in a Plane Poiseulle Flow
% Orr Sommerfeld Equation
% 03/25/13
% Bryan Kaiser

clear
clc
close all


% Neutral Stability Curve Loop set up

% 1) User Specified values ------------------------------
I = 10; % Number of iterations
% Vectors of important variables per iteration:
ar = zeros(1,I); % alpha real
dar = zeros(1,I);
wr = zeros(1,I); % omega real
dwr = zeros(1,I);
T1 = zeros(1,I);
T2 = zeros(1,I);
Numa = zeros(1,I);
Numw = zeros(1,I);
ar(1) = 2.04; % alpha = real component of the small disturbance
wavenumber
wr(1) = 0.81; % omega = real component of the small disturbance
frequency
Re = 7668; % Reynolds number based on channel height h - Does this
affect U(y)??????????
N = 400; % N = gridpoints. This parameter determines the location of
all
Damp = 1;
% cell bottom and top heights, starting at y = 0 and ending at y = 1.
e1 = 11;
e2 = 15;

% 2) Grid generation ------------------------------------------------
-------
[ dy G ] = GLgrid(N); % Function that creates a Gauss-Lobatto grid

% using the number of grid points specified above.
% G = locations of all y's, starting at 0 and ending at 1.
% dy = y difference from cell bottom to cell top
% plot(1:(N-1),dy)
% xlabel('Gridpoint')
% ylabel('dy')
% g = fliplr(G)
% plot(1:(N),g)
% xlabel('Gridpoint')
```

```
% ylabel('height')


% 3) Plane Poiseulle channel flow velocity U(y) profile ---------------
-------
% U(y) = 1 - 4(y-1/2)^2  (the profile for y = 0, y = 1 walls)
% U(y)'' = -8 (matches class notes, 2/12/13)
Uy = zeros(1,N-1); % Set up
for i = 1:N-1
    Uy(i) = (1.5-6*( (G(i+1)+G(i))/2-1/2)^2);
end
Uydp = -12; % U(y)''


% length(Uy)
% length(dy)
tic
% 4) Beginning to iterate alpha & omega
for j = 1:(I-1) % Number of iterations

    %Iteration = j

% 6) Equation matrix "A" set up ----------------------------------------
-------
n = 4;
A = zeros(n,n,N,N); % A Matrix populated with zeros only.


% A Matrices
A(:,:,1,1) = [ 1,0,0,0; 0,1,0,0; -2,0,-dy(1), 0; 0,-2,0,-dy(1) ]; % BC
for i = 1:N-2
    A(:,:,i+1,i+1) = [ -(ar(j))^2*dy(i), -dy(i), 2, 0;
Re*ar(j)*Uydp*dy(i)*sqrt(-1), (-dy(i))*((ar(j)^2)+(sqrt(-
1)).*Re*(ar(j)*Uy(i)-wr(j))) , 0, 2; -2, 0, -dy(i+1), 0;  0, -2, 0, -
dy(i+1) ];
end
A(:,:,N,N) = [ -(ar(j))^2*dy(N-1), -dy(N-1), 2, 0;
(Re*ar(j)*Uydp*dy(N-1)).*(sqrt(-1)), (-dy(N-1))*( (ar(j))^2+(sqrt(-
1))*Re*(ar(j)*Uy(N-1)-wr(j) )  ), 0, 2; 0, 0, 1, 0;  1, 0, 0, 0 ];

% B Matrices
for i = 1:N-1
    A(:,:,i+1,i) = [ -(ar(j))^2*dy(i), -dy(i), -2, 0;
Re*ar(j)*Uydp*dy(i)*sqrt(-1), (-dy(i))*((ar(j)^2)+(sqrt(-
1)).*Re*(ar(j)*Uy(i)-wr(j))),0,-2;  0,0,0,0;  0,0,0,0 ];
end
A(:,:,N,N-1) = [ -(ar(j))^2*dy(N-1), -dy(N-1), -2, 0;
Re*ar(j)*Uydp*dy(N-1)*(sqrt(-1)), (-dy(N-1))*((ar(j)^2)+(sqrt(-
1)).*Re*(ar(j)*Uy(N-1)-wr(j))),0,-2;  0,0,0,0;  0,0,0,0 ];

% Inner C Matrices
for i = 1:(N-1)
    A(:,:,i,i+1) = ([ 0,0,0,0; 0,0,0,0; 2,0,-dy(i),0; 0,2,0,-dy(i)]);
end
```

```
% 7) R Matrix Setup ------------------------------------------------
----
R = zeros(4,1,N,1);
R(2,1,1,1) = 1; % The top of R is y = 0, where s = 1, and  the bottom
is y = h.
Rs = R; % Saved value of original R, for later in the loop.


% 8) L U Factorization of A matrix ----------------------------------
----
[ L U ] = LUfactor(A);


% 9) Putting L & U into vector form ---------------------------------
----
S = size(L);
n = S(1);
N = S(3);
T = zeros(n,n,1,N-1);
for i = 1:N-1
    T(:,:,1,i) = (L(:,:,i+1,i)); % Vector of Lower diagonal of L
end
Ts = (T); % Saved value of original T.
D = zeros(n,n,1,N);
for i = 1:N
    D(:,:,1,i) = (U(:,:,i,i)); % Vector of Lower diagonal of U
end
Ds = (D); % Saved value of original D.
C = zeros(n,n,1,N-1);
for i = 1:N-1
    C(:,:,1,i) = (U(:,:,i,i+1)); % Vector of Upper diagonal of U
end
Cs = (C); % Saved value of original C.


% 10) Solving for delta (d) unknown column of vectors ----------------
----
% Creating the omega (w) vector of secondary unknown vectors
% Downwards sweep:
w = zeros(4,1,N,1);
w(2,1,1,1) = 1; % w0 = R0 (Notes 2/14)
for i = 1:N-1
    w(:,1,i+1,1) = (R(:,1,i+1,:))-(T(:,:,1,i)*w(:,1,i,:));
end
% Creating the delta (d) vector of primary unknown vectors
% Upwards sweep:
Li = [1:N];
L = fliplr(Li);
d = zeros(4,1,N,1);
d(:,1,N,1) = ((D(:,:,1,N))\w(:,1,N,1)); % dJ = (DJ^(-1))*wJ (Notes
2/14)
for i = 1:N-1
    d(:,1,L(i+1),:) = (D(:,:,1,L(i+1))\w(:,1,L(i+1),:))-
(D(:,:,1,L(i+1))\(C(:,:,1,L(i+1))*d(:,1,L(i),:)));
end
% The above loop: dj = (Dj^(-1))*(wj - cj*Dj+1), (Notes 2/14)
% The target variable f(y=h) => 0 :
```

```
T1(j) = (real(d(3,1,1))); % fr(y=h) real part of missing BC
T2(j) = (imag(d(3,1,1))); % fi(y=h) imag part of missing BC


% 11) Check to be inserted: A*d = R ? <--------------------------------
----
%[ Rcheck ] = LAPMulti( A,d );
% IsAns2 = find(Rcheck >= 1*10^(-e1))
% Checks: works properly: for N = 5, ans = 2 (s=1)


% 12) Partial Equation Matrices (dA/da and dA/dw)----------------------
----

% Equation matrix dAda set up
% Partial derivative of A with respect to alpha = dAda
n = 4;
dAda = zeros(n,n,N,N); % dAda Matrix populated with zeros only.
% with respect to alpha, in vector form (see line 43).
% Inner B Matrices of dAda
for i = 1:(N-1)
    dAda(:,:,i+1,i) = ([ -2*(ar(j))*dy(i),0,0,0; Re*Uydp*dy(i).*(sqrt(-
1)), -dy(i)*(2*(ar(j))+sqrt(-1).*Re*(Uy(i))) ,0,0; 0,0,0,0; 0,0,0,0]);
end
% Inner A Matrices of dAda
for i = 1:(N-1)
    dAda(:,:,i+1,i+1) = ([ -2*(ar(j))*dy(i),0,0,0; sqrt(-
1).*Re*Uydp*dy(i), -dy(i)*(2*(ar(j))+sqrt(-1).*Re*(Uy(i))) ,0,0;
0,0,0,0; 0,0,0,0]);
end

% Equation matrix dAdw set up
% Partial derivative of A with respect to omega = dAdw
n = 4;
dAdw = zeros(n,n,N,N); % dAdw Matrix populated with zeros only.
% Inner B Matrices of dAda
for i = 1:(N-1)
    dAdw(:,:,i+1,i) = ([0,0,0,0; 0,dy(i)*sqrt(-1)*Re,0,0; 0,0,0,0;
0,0,0,0]);
end
% Inner A Matrices of dAda
for i = 1:(N-1)
    dAdw(:,:,i+1,i+1) = ([0,0,0,0; 0,dy(i)*sqrt(-1).*Re,0,0; 0,0,0,0;
0,0,0,0]);
end


% 13) New R vector column (for dAda and dAdw)--------------------------
----
% Since A*(dd/dw) = -(dA/dw)*d and A*(dd/da) = -(dA/da)*d:
% Ra = -(dA/da)*d
% Rw = -(dA/dw)*d
% and A*(dd/da) = Ra and A*(dd/dw) = Rw.
Ra = LAPMulti( -dAda, d );
Rw = LAPMulti( -dAdw, d );
```

```
% 15) Solving for delta (d) unknown column of vectors for Ra & Rw  ----
----
% Creating the omega (w) vector of secondary unknown vectors
T = (Ts);
D = (Ds); % Checked, saved values work properly.
C = (Cs);
% Downwards sweep (matrix) up physically
w = zeros(4,1,N,1);
w(:,:,1,1) = (Ra(:,:,1,1)); % W0 = Ra0
for i = 1:N-1
    w(:,1,i+1,1) = (Ra(:,1,i+1,1))-(T(:,:,1,i)*w(:,1,i,:));
end
% Creating the delta (d) vector of primary unknown vectors
% Upwards sweep
Li = [1:N];
L = fliplr(Li);
ddda = zeros(4,1,N,1);
ddda(:,1,N,1) = (D(:,:,1,N)\w(:,1,N,1)); % Final d value
for i = 1:N-1
    ddda(:,1,L(i+1),:) = (D(:,:,1,L(i+1))\(w(:,1,L(i+1),:)))-
(D(:,:,1,L(i+1))\(C(:,:,1,L(i+1))*ddda(:,1,L(i),:)));
end
% Creating the omega (w) vector of secondary unknown vectors
% Downwards sweep
w = zeros(4,1,N,1);
w(:,:,1,1) = Rw(:,:,1,1); % W0 = Rw0
for i = 1:N-1
    w(:,1,i+1,1) = (Rw(:,1,i+1,:))-(T(:,:,1,i)*w(:,1,i,:));
end
% Creating the delta (d) vector of primary unknown vectors
% Upwards sweep
Li = [1:N];
L = fliplr(Li);
dddw = zeros(4,1,N,1);
dddw(:,1,N,1) = (D(:,:,1,N)\w(:,1,N,1)); % Final d value
for i = 1:N-1
    dddw(:,1,L(i+1),:) = (D(:,:,1,L(i+1)) \w(:,1,L(i+1),:))-
(D(:,:,1,L(i+1))\(C(:,:,1,L(i+1))*dddw(:,1,L(i),:)));
end


% 16) Check to be inserted: A*dd/dw = -dAdw*d and A*dd/da = Ra ? <-----
---------
% Rw;
% Rwtest = vpa(LAPMulti( A, dddw ));
% RwAns = find(Rw >= 1*10^(-e2));
% IsRwtestsame = find(Rwtest >= 1*10^(-e2));
% Before = Rw(:,:,N-1,1);
% After = Rwtest(:,:,N-1,1);
% RSsum = sum(Rw(RwAns(:)));
% RSsumtest = sum(Rwtest(IsRwtestsame(:)));
% Shouldbeone = RSsum/RSsumtest
% Very close for N = 20!


%%% CODE VERIFIED CORRECT UP TO THIS POINT %%%
```

```
% 17) Target variables at df/dw(y=0)and df/da(y=0) --------------------
----
dT1da = (real(ddda(3,:,1))); % dfr/da(y=0)
dT2da = (imag(ddda(3,:,1))); % dfi/da(y=0)
dT1dw = (real(dddw(3,:,1))); % dfr/dw(y=0)
dT2dw = (imag(dddw(3,:,1))); % dfi/dw(y=0)
% T1(j) = vpa(real(d(3,1,1)),Z); % fr(y=h) real part of missing BC
% T2(j) = vpa(imag(d(3,1,1)),Z); % fi(y=h) imag part of missing BC

% 18) Calulation of next guess for alpha and omega --------------------
----
Den = (dT1da*dT2dw-dT2da*dT1dw);

Numa(j) = (T2(j)*dT1dw-T1(j)*dT2dw);

dar(j) = (Numa(j)/Den); % Change in alpha for next iteration
dara = abs(dar);

Numw(j) = (T1(j)*dT2da-T2(j)*dT1da);

dwr(j) = (Numw(j)/Den); % Change in omega for next iteration
dwra = abs(dwr);

if dara(j) <= 10^(-8)
    if dwr(j) <= 10^(-8)
        formatSpec = 'At iteration %2.0f, convergence requirement met
\n';
        fprintf(formatSpec,j)
    end
end

% 19) The next guess:
ar(j+1) = (ar(j) + Damp*dar(j));
wr(j+1) = (wr(j) + Damp*dwr(j));
%pause

end
toc

% 20) Alpha Results
dara = abs(dar);
dwra = abs(dwr);
Convaloc = find(dara <= 10^(-8)); % Convergence requirement
Convwloc = find(dwra <= 10^(-8)); % Convergence requirement
Conva = dar(Convaloc(1)); % Converged requirement value
Convw = dwr(Convwloc(1)); % Converged requirement value

formatSpec = 'For iteration %2.0f and Re = %2.0f, alpha %4.3f and omega
%4.3f \n';
fprintf(formatSpec,I(end),Re, ar(end),wr(end))
formatSpec = 'At iteration %2.0f, alpha convergence and omega
convergence below 10^(-8) \n';
fprintf(formatSpec,Convaloc(1))
```

```
format Long
Result_Vector = [ 20; ar(end); wr(end); Convaloc(1) ]

% 21) Plots

% figure
% set(gca,'FontName','Times New Roman','Fontsize',16)
% p1 = plot(1:I,dar);
% xlabel('Iteration')
% ylabel('d alpha')
%
% figure
% set(gca,'FontName','Times New Roman','Fontsize',16)
% p2 = plot(1:I,dwr);
% xlabel('Iteration')
% ylabel('d omega')
%

% figure
% eta = G;
% del = d;
%
%   for j=1:N
%     phi(j)=del(1,1,j,1);
%     s(j)=del(2,1,j,1);
%     f(j)=del(3,1,j,1);
%     g(j)=del(4,1,j,1);
% end
% %real part
% subplot(1,4,1)
% plot(real(phi),eta,':',imag(phi),eta,'--',abs(phi),eta);
% grid on;
% xlabel('\phi');
% ylabel('\eta');
% legend('Real','Imaginary','Magnitude');
% subplot(1,4,2)
% plot(real(s),eta,':',imag(s),eta,'--',abs(s),eta);
% grid on;
% xlabel('s');
% ylabel('\eta')
% legend('Real','Imaginary','Magnitude');
% subplot(1,4,3)
% plot(real(f),eta,':',imag(f),eta,'--',abs(f),eta);
% grid on;
% xlabel('f');
% ylabel('\eta');
% legend('Real','Imaginary','Magnitude');
% subplot(1,4,4)
% plot(real(g),eta,':',imag(g),eta,'--',abs(g),eta);
% grid on;
% xlabel('g');
% ylabel('\eta');
% legend('Real','Imaginary','Magnitude');
```

2) Gauss-Lobatto Grid Generation, function

```
function [ dy G1 ] = GLgrid( n )
% Generates a grid of Gauss Lobatto distribution

% N = Number of grid points
N = n;

% Generating the Gauss-Lobatto grid points
G1(1:N) = (0.5+cos((0:N-1)*pi/(N - 1))/2); % Location of y, starting at
0
% and ending at 1.


% Vector dy (center points = N - 1)
dy = abs(diff(G1)); % y difference from cell bottom to cell top

% Test plot
% close all
% Y1 = linspace(0,1,N-1);
% plot(dy,Y1)
% title('Gauss Lobatto Grid')
% xlabel('Cell Center Height (dy)')
% ylabel('Channel Height')

end
```

3) Linear Algebraic Product of Matrices of Matrices or Vectors of Vectors, function

```
function [ C ] = LAPMulti( A,B)
% Linear Algebraic Product (LAP) of two multidimensional matrics of
equal
% dimensions
% A = Matrix of matrices (all square)
% B = Matrix of matrices (all square) OR(!) vector of vectors

Asize = size(A);
nA = Asize(1); % Minor rows in A
mA = nA; % Square minor matrix, minor cols
NA = Asize(3); % Major rows in A
MA = NA; % Square major matrix, major cols

A2 = vpa(A,1000);
A = vpa(A2,1000);
B2 = vpa(B,1000);
B = vpa(B2,1000);

if NA ~= MA
    fprintf('A is not square')
end
```

```
Bsize = size(B);
Temp = length(Bsize);

if Temp == 3 % B is a vector
    %fprintf('B is a vector')
    nB = Bsize(1); % Minor rows in B
    mB = Bsize(2); % Minor cols in B (should be 1)
    if mB ~= 1
        fprintf('Problem with vector dimensions, check it')
    end
    NB = Bsize(3); % Major rows in B (must equal cols of A)
    MB = 1; % One major column in B (must be 1)

    % Insert matrix dimension check: NB = MA
    N = NB;
    Ci = zeros(nB,mB,N,1);
    for j = 1:N % For each row
        k = 1; % For each column
            E = zeros(nB,mB,1,N);
            Itr = j;
            Itc = k;
            for i = 1:N
                E(:,:,1,i) = (A(:,:,j,i)*B(:,:,i,k));
            end
            Ci(:,:,j,k) = (SumMultiArray(E));

    end
    C = (Ci);



elseif Temp == 4 % B is a matrix
    %fprintf('B is a matrix')
    nB = Bsize(1); % Minor rows in B
    mB = Bsize(2); % Minor cols in B
    NB = Bsize(3); % Major rows in B
    MB = Bsize(4); % Major cols in B

    % Insert matrix dimension check: NB = MB = NA = MB
    N = MB;
    Ci = zeros(nB,mB,N,N);
    for j = 1:N % For each row
        for k = 1:N % For each column
            E = zeros(nB,mB,1,N);
            Itr = j;
            Itc = k;
            for i = 1:N
                E(:,:,1,i) = (A(:,:,j,i)*B(:,:,i,k));
            end
            Ci(:,:,j,k) = (SumMultiArray(E));
        end
    end
    C = (Ci);
```

```
else
    fprintf('Error in LAPMatMat due to column sizing, check it')
end

end
```

4) Lower and Upper Factorization of Matrices of Matrices, function

```
function [ L U ] = LUfactor( A )
% LU factorization
% A must be a matrix of matrices

Temp = size(A);
n = Temp(1); % Number of scalar points in sub-matrices
N = Temp(3); % Number of sub-matrices in A matrix


A2 = (A);
A = (A2);

% Matrices of split tridiagonal
T1 = eye(n,n);
L = zeros(n,n,N,N);
for i = 1:N
    L(:,:,i,i) = (T1); % L matrix set up, with I matrices
end
L(:,:,2,1) = (A(:,:,2,1)/A(:,:,1,1)); % T1 = B1/A0 = B1/D0


U = zeros(n,n,N,N); % U matrix set up, all zeros
U(:,:,1,1) = (A(:,:,1,1)); % Delta(0,0) = A0
for i = 1:N-1
    U(:,:,i,i+1) = (A(:,:,i,i+1)); % C0 through CJ
end


for i = 2:N-1
    U(:,:,i,i)= (A(:,:,i,i)) - (L(:,:,i,i-1))*(A(:,:,i-1,i)); % Filling
out D values, 1-(J-1), starting at the second row L(:,:,i,i-1),Z
A(:,:,i,i-1)/A(:,:,i-1,i-1),Z
    L(:,:,i+1,i)= (A(:,:,i+1,i)/U(:,:,i,i)); % Filling out T values,
T2-TJ started at the third row
end
%whos L

U(:,:,N,N) = A(:,:,N,N)-L(:,:,N,N-1)*A(:,:,N-1,N); % DJ

end
```

5) Summation of Matrices of Matrices, function

```
function [ C ] = SumMultiArray( A )
```

```
% Sum of Multidimensional Array across rows
% Input "A" Must be a single major row of a multidimensional array
% i.e. input: A(1:n,1:m,1,1:N)

% N = Ni; % major dimension
% M = 1;


S = size(A);
n = S(1);
m = S(2);
M = S(3); % Should equal 1
N = S(4);

A2 = (A);
A = (A2);

if M ~= 1
    fprintf('Error in SumMultiArray function, check it')
end

Cii = zeros(n,m);
for k = 1:n
    for j = 1:m

    Ci = zeros(1,N);
        for i = 1:N
            Ci(i) = (A(k,j,1,i)); % A(loc,loc,1,vector to be summed)
        end

    Cii(k,j) = (sum(Ci));
    end
end
C = (Cii);
% Output: C(1:n,1:m,1,1)

end
```