OCE680 Assignment 5, Instructor's notes

1: The fourth derivative matrix [10]

(**a**) Here is the 2nd-order, centered finite-difference approximation to the fourth derivative:

$$f_i^{(4)} = f_{i-2} - 4f_{i-1} + 6f_i - 4f_{i+1} + f_{i+2}.$$

(**b**) Boundary conditions are implemented as follows:

- for rigid boundaries,

$$f_1^{(4)} = 7f_1 - 4f_2 + 7f_3 \; ; \quad f_N^{(4)} = f_{N-2} - 4f_{N-1} + 7f_N,$$

- for frictionless boundaries,

$$f_1^{(4)} = 5f_1 - 4f_2 + 7f_3 \; ; \quad f_N^{(4)} = f_{N-2} - 4f_{N-1} + 5f_N,$$

- and for both boundary types,

$$f_2^{(4)} = -4f_1 + 6f_2 - 4f_3 + f_4 \; ; \quad f_{N-1}^{(4)} = f_{N-3} - 4f_{N-2} + 6f_{N-1} - 4f_N.$$

2: Matrix solution of the Orr-Sommerfeld equation [15]

(**a**) My code to solve the Orr-Sommerfeld equation is given below.

(**b**) For the given length and velocity scales, the wavelength is $2\pi/1.55 * 15m = 61m$ and the e-folding time is $1/.0152 * 15/2 = 493s$.

3: Wave resonance in a jet [5]

See figure 16.27.

4: A convectively unstable layer in an inviscid fluid, revisited. [10]

The stability equation is (cf. 2.3.10):

$$\sigma^2 \Big(\frac{\mathrm{d}^2}{\mathrm{d}z^2} - \tilde{k}^2 \Big) \hat{w} = \tilde{k}^2 B_z \hat{w} \; , \tag{16.1.14}$$

where

$$B_z = B_{z0}(1 - 2\mathrm{sech}^2 \alpha z).$$

How do you choose values for the constants B_{z0} and α? That problem is sidestepped by scaling. Suppose we have a solution procedure

$$\sigma = \mathscr{F}(z, B_z, \tilde{k})$$

Define scaled variables

$$\sigma = \sqrt{B_{z0}}\sigma_* \; ; \; z = z_*/\alpha \; ; \; \tilde{k} = \alpha \tilde{k}_* \; ; \; B_z = B_{z0}\beta \; .$$

Substitute and get

$$\sigma_*^2 \Big(\frac{\mathrm{d}^2}{\mathrm{d}z_*^2} - \tilde{k}_*^2 \Big) \hat{w} = \tilde{k}_*^2 \beta(z_*) \hat{w},$$

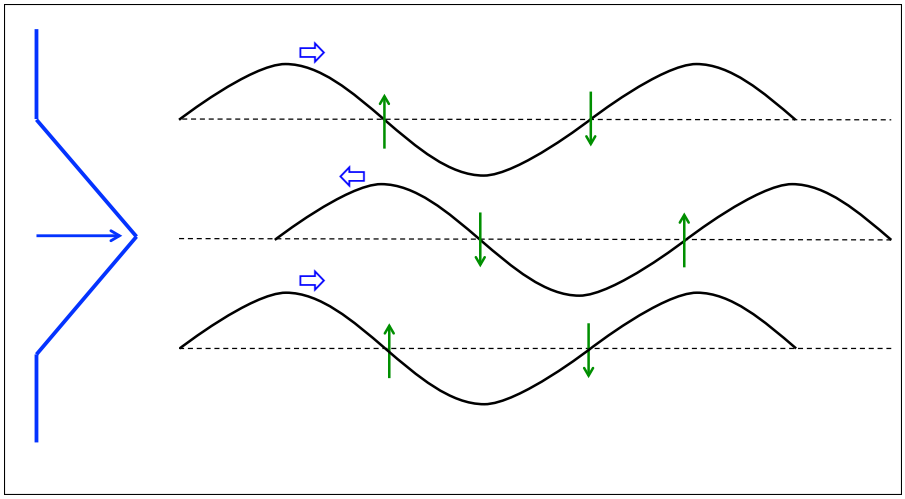

Figure 16.27: Schematic velocity profile for a triangular jet showing three vorticity waves that resonate to form the sinuous mode. The waves move at a common speed between zero and the maximum speed of the jet. Relative to the surrounding flow, the upper and lower waves propagate to the right; the middle wave to the left.

with

$$\beta = 1 - 2\text{sech}^2 z_*.$$

This is isomorphic to (16.1.14), and therefore the solution procedure is

$$\sigma_* = \mathscr{F}(z_*, \beta, \tilde{k}_*) \ .$$

The parameters B_{z0} and α have been removed from the problem.

(a) The growth rate curve increases monotonically (figure 16.28), approaching a limit as $k_* \to \infty$. This is similar to convection in an inviscid fluid with uniform B_z. The limiting growth rate is 1, or $\sigma = \sqrt{B_{z0}}$, consistent with the upper bound found in section 2.3.3.

(b) The results match (figure 16.29).

(c) The case done previously does *not* represent the fastest-growing mode.

(d) The fastest-growing mode is at $k_* = \infty$. (I did not anticipate this result when I posed the question.) If you chose a reasonably large value of k_*, your eigenfunction was sharply peaked at $z_* = 0$.

```
function [sig,w]=VSF(z,U,nu,k,l,bc,imode)
%
% USAGE: [sig,w, vort]=VSF(z,U,nu,k,l,iBC1,iBCN)
% Stability analysis for a viscous, parallel shear flow
%
% INPUTS:
% z = vertical coordinate vector (evenly spaced)
% U = velocity profile
% nu = viscosity
% k,l = x, y wavenumbers (default l=0)
% bc = character array [bc(1) bc(2)] to specify bcs at z(0), z(N+1).
%     r: rigid [default]
```

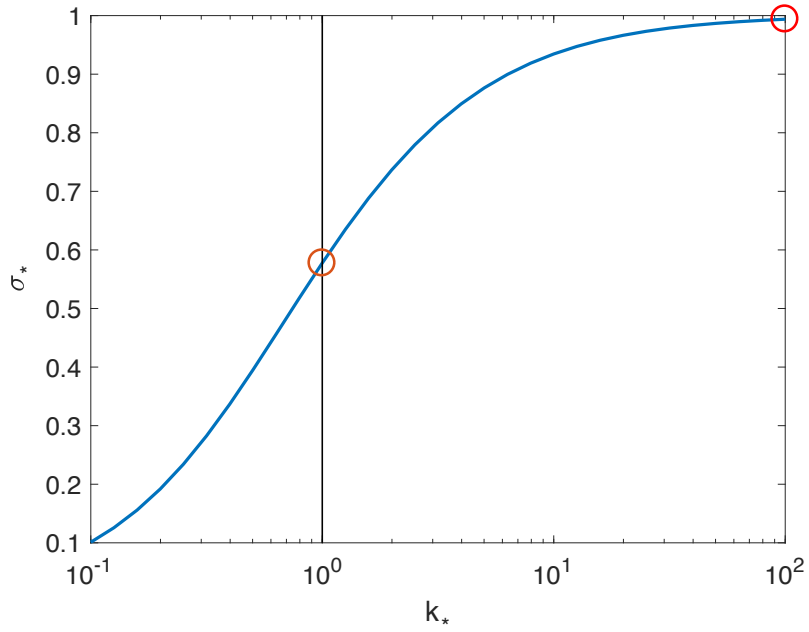

Figure 16.28: Scaled growth rate versus wavenumber for unstable layer. Circles show the analytical solution for $k_* = 1$ and the fastest-growing mode among the wavenumbers scanned.

```
%     f: frictionless
% imode = mode choice (default imode=1)
%
% OUTPUTS:
% sig = growth rate of FGM
% w = vertical velocity eigenfunction
%
% W. Smyth, Feb16

% check for equal spacing
if abs(std(diff(z))/mean(diff(z)))>.000001
    disp(['VSF: values not evenly spaced!'])
    sig=NaN;
    return
end

% defaults
if ~exist('l');l=0;end
if(~exist('bc'))
    bc='rr';
end
if ~exist('imode');imode=1;end

% define constants
ii=sqrt(-1);
```

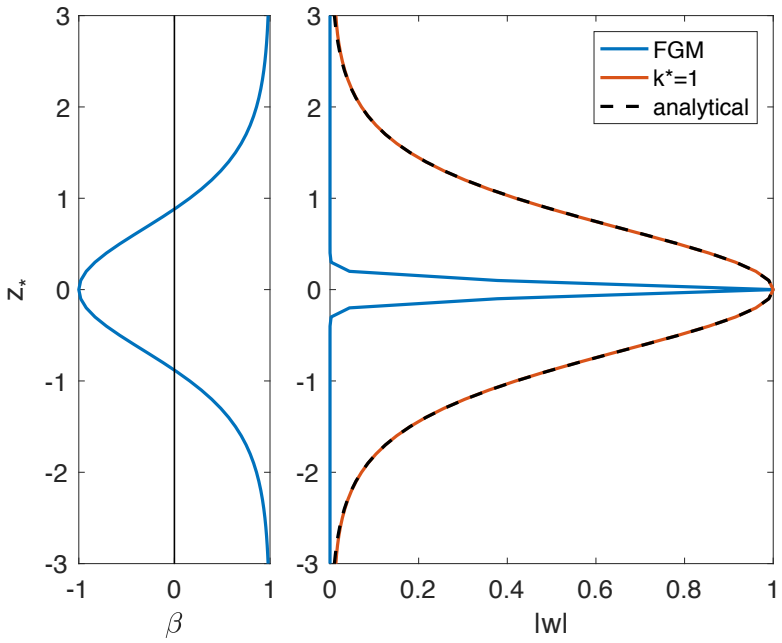

Figure 16.29: Left: scaled stratification profile. Right: eigenfunctions as indicated. Note that the eigenfunctions are real.

```
del=mean(diff(z));N=length(z);
kt=sqrt(k^2+l^2);

Uzz=ddz2(z)*U;

% Second derivative matrix
D2=ddz2(z);
% impermeable
D2(1,:)=0;D2(1,1:2)=[-2 1]/del^2;
D2(N,:)=0;D2(N,N-1:N)=[1 -2]/del^2;

% Fourth derivative matrix
D4=ddz4(z);
if bc(1)=='f';    % frictionless
    D4(1,:)=0;D4(1,1:3)=[5 -4 1]/del^4;
elseif bc(1)=='r';     % rigid
    D4(1,:)=0;D4(1,1:3)=[7 -4 1]/del^4;
else
    display(['VSF: Boundary condition 1 not understood'])
    sig=nan;
    return
end
D4(2,:)=0;D4(2,1:4)=[-4 6 -4 1]/del^4;
if bc(2)=='f'; % frictionless
    D4(N,:)=0;D4(N,N-2:N)=[1 -4 5]/del^4;
elseif bc(2)=='r'
```

```
    D4(N,:)=0;D4(N,N-2:N)=[1 -4 7]/del^4;
else
    display(['VSF: Boundary condition 2 not understood'])
    sig=nan;
    return
end
D4(N-1,:)=0;D4(N-1,N-3:N)=[1 -4 6 -4]/del^4;



% Set up arrays for eigenvalue analysis
Id=eye(N);
A=D2-kt^2*Id;
B=-ii*k*diag(U)*A+ii*k*diag(Uzz)+nu*(D4-2*kt^2*D2+kt^4*Id);

% Solve generalized eigenvalue problem
[w_hat,e]=eig(B,A);sigma=diag(e);

% Sort eigvals & eigvecs
[sr,ind]=sort(real(sigma),1,'descend');
sigma=sigma(ind);
w_hat=w_hat(:,ind);

% Output selected growth rate, vertical velocity and spanwise vorticity
sig=sigma(imode);
w=w_hat(:,imode);

return
end
```

And here is my function to solve the general equation for convection:

```
function [sig1,w1]=convect(z,Bz,kt,BC,nmode)
%
% MINIMAL USAGE: [sig]=convect(z,Bz,kt)
%    FULL USAGE:  [sig,w]=convect(z,Bz,kt,nmode)
%
% Stability analysis for inviscid, homogeneous, parallel shear flow
% (Rayleigh equation)
%
% INPUTS:
% z = vertical coordinate vector (evenly spaced)
% Bz = buoyancy gradient profile
% kt = wave vector magnitude
% BC = vector of boundary conditions at z(1) and z(end):
%       1 = impermeable
%       2 = asymptotic
% nmode = mode selection in terms of growth rate:
%       1 = fastest-growing mode [default nmode=1]
%       2 = second-fastest mode, etc.
```

```
%
% OUTPUTS:
% sig = complex growth rate
% w = vertical velocity eigfn
%
% W. Smyth, Feb 2018

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 1: Preliminaries
%

% check for equal spacing
if abs(std(diff(z))/mean(diff(z)))>.000001
    disp(['ddz2: values not evenly spaced!'])
    sig1=NaN;
    return
end

% defaults
if nargin<4; iBC=[1 1];end  % impermeable BCs
if nargin<5; nmode=1;end    % choose FGM

% define constants
del=z(2)-z(1);
N=length(z);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 2: Set up derivative matrices
%
D2=ddz2(z); % 2nd derivative matrix with 1-sided boundary terms

% Boundary conditions
% Impermeable boundaries
D2(1,:)=0;D2(1,1)=-2/del^2;D2(1,2)=1/del^2;
D2(N,:)=0;D2(N,N)=-2/del^2;D2(N,N-1)=1/del^2;
% Change to asymptotic boundaries if requested
if BC(1)==2
        D2(1,:)=0;D2(1,1)=-2*(1+del*kt)/del^2;D2(1,2)=2/del^2;
end
if BC(2)==2
    D2(N,:)=0;D2(N,N)=-2*(1+del*kt)/del^2;D2(N,N-1)=2/del^2;
end
% 2nd derivative matrix complete

% Laplacian matrix
L=D2-kt^2*eye(N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 3: Set up stability matrices,
```

```
%          solve eigval problem, sort results
%

% Set up arrays for eigenvalue analysis.
A=L;
B=kt^2*diag(Bz);

%stop
% Solve eigenvalue problem.
[w,S]=eig(B,A);
s=sqrt(diag(S));

% Sort eigvals and eigvecs by real growth rate
[sr,ind]=sort(real(s),1,'descend');
sigma=s(ind);
w=w(:,ind);

% Save the mode selected via nmode
sig1=sigma(nmode);w1=w(:,nmode);

% Normalize by the value at the max of abs(w). This works better than using
% the value at z=0, which may turn out to be zero.
cnorm=w1(abs(w1)==max(abs(w1)));
w1=w1/cnorm;

return
end
```