14: Matrix solution of the Orr-Sommerfeld equation

(**a**) A subroutine 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 $1/.0152 * 15/2 = 493s$.

```
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]
%     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);
del=mean(diff(z));N=length(z);
kt=sqrt(k^2+l^2);
Uzz=ddz2(z)*U;
```

```
% Second derivative matrix
D2=ddz2(z);
% impermeable bc
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 and vertical velocity
sig=sigma(imode);
w=w_hat(:,imode);
return
end
```