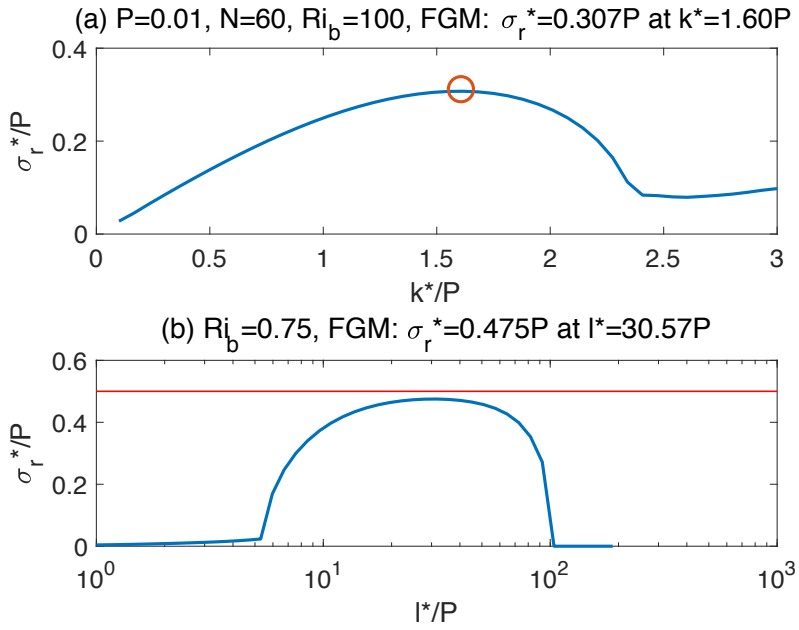

Figure 1: Scaled growth rate versus wavenumber for the agestrophic Eady problem: (upper) baroclinic modes with $\ell^\star = 0$, (lower) symmetric modes with $k^\star = 0$. Low resolution version.

20: Instabilities of the Eady model

We use the methods described in 8.8, including the shear scaling (section 8.8.5). The vertical range is $z^\star = [-1/2, 1/2]$, and is discretized using $N = 60$ grid points. In addition, we find it convenient to scale wavenumbers and growth rates by the Prandtl ratio P.

(a) With $\ell^\star = 0$ (baroclinic modes), the maximum growth rate is $\sigma^\star = 0.307P$ (figure 1a), which compares well with the analytical result $0.310P$ (see equation 8.61).

(b) With $k^\star = 0$ (symmetric modes), the computed growth rate is not independent of wavenumber, as is the analytical solution, but rather increases to a maximum value of $0.475P$ when $\ell^\star = 30.6P$, then returns to zero at higher $\ell^\star$ (figure 1b). When N is increased to 300 (figure 2b), the maximum growth rate increases to $0.495P$, much closer to the analytical value 1/2. Also, the range of nonzero growth rates extends upward. This suggests that the discrepancy with the analytical solution is partly a resolution issue. Note, though, that the lower limit of the unstable range does not change with increasing N. Evidently this low wavenumber cutoff is due to the boundaries, which are absent in the analytical solution.

(c) Using 180 grid points and $P = 0.01$, we find that baroclinic ($\ell = 0$) and symmetric ($k = 0$) instabilities have equal growth rates, $\sigma^\star = 0.22P$, when $Ri = 0.94$. For $Ri < 0.94$, the symmetric mode dominates.

Here is our subroutine for solving (8.77):

```
function [sig,xi,w,b]=baro(z,Uz,tz,f,k,l,imode)
%
% USAGE: [sig,xi,w,b]=baro(z,Uz,tz,f,k,l,imode)
% Stability analysis for a flow in thermal wind balance
%
% INPUTS:
```

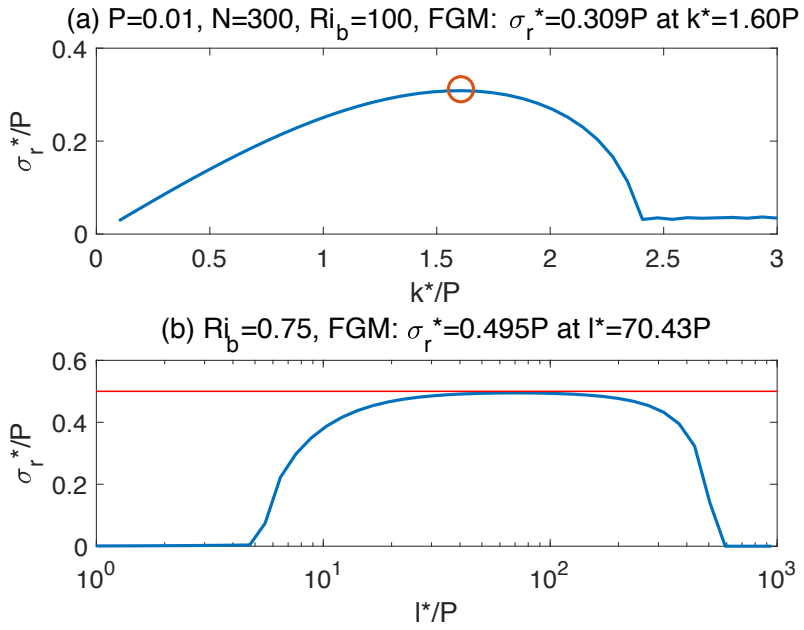

Figure 2: Scaled growth rate versus wavenumber for the agestrophic Eady problem: (upper) baroclinic modes with $\ell\star = 0$, (lower) symmetric modes with $k^\star = 0$. High resolution version.

```
% z = vertical coordinate vector (evenly spaced)
% Uz = shear (constant); code assumes that U=Uz*z;
% tz = vertical buoyancy gradient, can be a single value or a profile
% k,l = wave vector
% imode = mode choice (default imode=1)
%         imode=0: output all modes, sorted by growth rate
%
% OUTPUTS:
% sig = growth rate of FGM
% eigenfunctions (optional):
% xi = vertical vorticity
% w = vertical velocity
% b = buoyancy
%
% CALLS:
% ddz,ddz2
%
% W. Smyth, Oregon State University, March 2016

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

% check for equal spacing
if abs(std(diff(z))/mean(diff(z)))>.000001
```

```
    disp(['baro: values not evenly spaced!'])
    sig=NaN;
    return
end

% defaults
if ~exist('imode');imode=1;end

% define constants
ii=complex(0.,1.);
del=mean(diff(z));N=length(z);
kt=sqrt(k^2+l^2);

U=Uz*z; % compute U from uniform shear;
By=-f*Uz; % compute By from thermal wind balance

% expand tz into a vector if necessary
if length(tz)==1;
    tz=tz*ones(size(z));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 2: Derivative matrices and BCs

% D2: 2nd derivative for w (impermeable boundary)
D2=ddz2(z);
D2(1,:)=0;D2(1,1:2)=[-2 1]/del^2;
D2(N,:)=0;D2(N,N-1:N)=[1 -2]/del^2;

% D1w: 1st derivative for w
D1w=ddz(z);
D1w(1,:)=0; D1w(1,2)=1/(2*del); % impermeable
D1w(N,:)=0; D1w(N,N-1)=-1/(2*del);

% D1z: 1st derivative for xi
D1z=ddz(z);
D1z(1,:)=0;
D1z(N,:)=0;
D1z(1,[1 2])=[-1 1]*2/(3*del);
D1z(N,[N-1 N]) =[-1 1]*2/(3*del);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 3: Assemble stability matrices A and B
%
Id=eye(N);
L=D2-kt^2*Id;

A=[Id Id*0 Id*0 ; Id*0 L Id*0; Id*0 Id*0 Id];
```

```
b11=-ii*k*diag(U);
b12=ii*l*Uz*Id+f*D1w;
b13=Id*0;

b21=-f*D1z;
b22=-ii*k*diag(U)*L;
b23=-Id;

b31=ii*k*By*Id;
b32=-ii*l*By*D1w-kt^2*diag(tz);
b33=-ii*k*diag(U);

B=[b11 b12 b13; b21 b22 b23; b31 b32 b33];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 4: Solve eigenvalue problem and extract results
%
% Solve generalized eigenvalue problem

if nargout==1 % compute eigvals only (faster)
    sigma=eig(B,A);
    [sr,ind]=sort(real(sigma),1,'descend');
    sigma=sigma(ind);
    if imode==0
        sig=sigma;
    elseif imode>0
        sig=sigma(imode);
    end
elseif nargout>1 % compute eigvals and eigvecs
    [v,e]=eig(B,A);sigma=diag(e);
    [sr,ind]=sort(real(sigma),1,'descend');
    sigma=sigma(ind);
    v=v(:,ind);
    if imode==0
        sig=sigma;
        xi=v(1:N,:);
        w=v(N+1:2*N,:);
        b=v(2*N+1:3*N,:)/kt^2;
    elseif imode>0
        sig=sigma(imode);
        xi=v(1:N,imode);
        w=v(N+1:2*N,imode);
        b=v(2*N+1:3*N,imode)/kt^2;
    end
end
return
```

And here is the code we used to do the calculations for this exercise:

```
% Stability analysis of Eady model using non-QG code
% W. Smyth 2018

%% preliminaries
clear
close all
fs=18;
ms=14;
lw=2;

%% set parameters,
P=.01;
Rib1=100;
Rib2=.75;
Nk=45;
Nl=45;

% define vertical grid
N=300;
z_st=linspace(-1/2,1/2,N+1)';z_st(1)=[];z_st(end)=[];
dz=mean(diff(z_st))

fac=P; % plots will be scaled using P

%% loop over k

k_sts=linspace(.1,3,Nk)*P;
for i=1:Nk
    disp(num2str(i/Nk));
    k_st=k_sts(i);
    [sig_k(i)]=baro(z_st,-1,Rib1,P*sqrt(Rib1),k_st,0);
end
% identify fastest-growing mode
[sig_k_mx,i_k_mx]=max(real(sig_k));
% refine fastest-growing mode using quadratic fit
ss=real(sig_k(i_k_mx+[-1 0 1]))
kk=k_sts(i_k_mx+[-1 0 1])
p=polyfit(kk-kk(2),ss,2);
kmx=kk(2)-.5*p(2)/p(1)
sig_kmx=baro(z_st,-1,Rib1,P*sqrt(Rib1),kmx,0)

%% loop over l
l_min=1;
l_max=pi/dz;
l_sts=P*10.^linspace(log10(l_min),log10(l_max),Nl);
for i=1:Nl;
```

```
    disp(num2str(i/Nl));
    l_st=l_sts(i);
    [sig_l(i)]=baro(z_st,-1,Rib2,P*sqrt(Rib2),0,l_st);
end
% identify fastest-growing mode
[sig_l_mx,i_l_mx]=max(real(sig_l));
% refine fastest-growing mode using quadratic fit
ss=real(sig_l(i_l_mx+[-1 0 1]))
ll=log10(l_sts(i_l_mx+[-1 0 1]))
p=polyfit(ll-ll(2),ss,2);
llmx=ll(2)-.5*p(2)/p(1);
lmx=10^llmx;
sig_lmx=baro(z_st,-1,Rib2,P*sqrt(Rib2),0,lmx)


%% plot results

figure
% l=0 (baroclinic modes)
subplot(2,1,1)
% plot numerical eigenvalues for l*=0
leg1(1)=plot(k_sts/fac,real(sig_k)/fac,'linewidth',lw)
hold on
title(sprintf('(a) P=%.2f, N=%.0f, Ri_b=%.0f, FGM: \\sigma_r*=%.3fP at k*=%.2fP',P,N,Rib1,sig_k
ylabel('\sigma_r*/P','fontsize',fs)
set(gca,'fontsize',fs-2)
xlabel('k*/P','fontsize',fs)
set(gca,'fontsize',fs-2)

% add theoretical QG value
kmax_QG=1.61*P;
sigmax_QG=0.31*P;
plot(kmax_QG/fac,sigmax_QG/fac,'o','markersize',16,'linewidth',lw)

% k=0 (symmetric modes)
subplot(2,1,2)
% plot numerical eigenvalues
leg1(2)=semilogx(l_sts/fac,real(sig_l)/fac,'linewidth',lw) % numerical eigenvalues
hold on
ylabel('\sigma_r*/P','fontsize',fs)

set(gca,'fontsize',fs-2)
xlabel('l*/P','fontsize',fs)
set(gca,'fontsize',fs-2)

smax_l=max(real(sig_l))/fac;
title(sprintf('(b) Ri_b=%.2f, FGM: \\sigma_r*=%.3fP at l*=%.2fP',Rib2,sig_lmx/P,lmx/P),'fontsiz
```

```
% add theoretical value
sUzP_theo=sqrt(1-Rib2);
if Rib2>1; sUzP_theo=0; end
semilogx(xlim,sUzP_theo*[1 1],'r','linewidth',1)
xlim([1 1000])
return
```