Electronic Supplement

A) Matlab code for analytical solution

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PropMatrixTau.m - calculates the growth rate of a Rayleigh-Taylor
instability using
%% the Propagator matrix method (see Hager&O'Connell JGR 1981).
%% The methods solves Du=Au+b, where D = d/dz, u = [Vz, Vx, Tzz, Txz].
%% A comes from the equations of motion and the constitutive relations.
%% b is a forces vector, usually containing [0 0 rho.g.w 0] for the
%% density jump at each interface between layers.
%% We define P(A,z,z0) = exp(A*(z-z0)), and then use, for a two-layer-
confined setup:
%%    u(top) = P2*u(above_interface), u(under_interface) = P1*u(bottom)
%%    u(above_interface) = u(under_interface) + b(at_interface) -->
                  u(top) = P2*P1*u(bottom) + P2*b(at_interface)
%% Note that this code uses Matlab's Symbolic toolbox.
%%
%% Written by: Einat Lev, MIT, 2007.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta = input('What is delta?');
disp ('Please select the type of the upper layer 1=isotropic, 2=dipping
anisotropy, 3=horizontal anisotropy, 4 = isotropic, low viscosity')
configuration = input('Configuration=')

syms h k d y y0 Tzz_b Txz_b rgw Tzz_t Txz_t A B

%define range of wave numbers
q = [0.01:0.05:4 5:3:100];

Aiso = ...
    [[0 -k       0  0  ] ;
    [k  0       0 1/h ] ;
    [0  0       0 -k  ] ;
    [0 4*h*k^2  k  0  ]];

%Horizontal axis:
Aani_horiz = ...
    [[0 -k       0  0  ] ; ...
     [k  0       0 1/d] ; % should be 1/d/h, but we assume h=1
     [0  0       0 -k  ] ;
     [0 4*k^2  k  0  ]]; % should be 4hk^2

%45-degrees dipping axis:
Aani_45 = ...
   [[0 -k          0  0  ] ; ...
    [k  0          0 1/h] ;
    [0  0          0 -k  ] ;
    [0 4*d*k^2   k  0  ]]; % should be 4dhk^2

Aiso_reducedEta = ...
   [[0 -k          0  0  ] ; ...
    [k  0          0 1/h/d] ;
    [0  0          0 -k  ] ;
    [0 4*d*h*k^2   k  0  ]]; % should be 4dhk^2

Aani_45 = subs(Aani_45, 'd', delta);
Aani_horiz = subs(Aani_horiz, 'd', delta);
```

```
%% Define the propagator matrices. Note: usually the propagator matrix
is exp(A*y-y0),
% % but here we know that y-y0 is always 1, so we ignore it to make the
calculation faster

%% Calculate propagator matrix for each A axis:
Piso        = expm(Aiso);        Piso       = subs(Piso,'h', 1);
Pani_horiz  = expm(Aani_horiz);  Pani_horiz = subs(Pani_horiz,'h', 1);
Pani_45     = expm(Aani_45);     Pani_45    = subs(Pani_45, 'h', 1);
Piso_reducedEta = expm(Aiso_reducedEta); Piso_reducedEta =
subs(Piso_reducedEta, 'h', 0.3162); %0.55 for (1+0.1)/2, or
0.3162=sqrt(1*0.1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%set properties of layers:
Plower = Piso;

switch configuration
    case 1,
        Pupper = Piso;
    case 2,
        Pupper = Pani_45;
    case 3,
        Pupper = Pani_horiz;
    case 4,
        Pupper = Piso_reducedEta;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%Set the boundary conditions to be no-slip at the top and bottom
U_bottom = transpose([0 0 0 0]); %Infinite half space
U_underIF = transpose([A/2/k, B/2/k, A, B]); %checked that when
propagated downwards to -Infinity, all components die.
U_overIF = U_underIF + transpose([0 0 1 0]);
U_top = Pupper*U_overIF;

%%  Slove system of equation using
% %  Utop = [0 0 Tzzt Txzt]', Utop=Utop1*[Txzb Tzzb Txzt Tzzt]' + rgw*b
,
% %  B=Utop1+[0 0 -1 -1]*[Txzb Tzzb Tzzt Txzt]'
%%
%%        --> B*[Txzb Tzzb Txzt Tzzt]' = rgw*b
S = solve (U_top(1), U_top(2), 'A,B');

disp('Now caculating U_underIF...')
U_underIF = transpose([S.A/2/k, S.B/2/k, S.A, S.B]);
U_underIF_beforeDeltaSub = U_underIF;
U_underIF = subs(U_underIF, 'd', delta);

disp('Now calculating U_overIF...')
U_overIF = U_underIF + transpose ([0 0 1 0]); %should normally be: +
transpose([0 0 rgw 0]);

disp('Now taking the vertical velocity from U_overIF...')
V = simplify(U_overIF(1));
```

```
disp('Now substituting rgw=1 in V ...')
V_rgw = subs(V, 'rgw', 1);
Tau = -1/V_rgw;

disp('Now calculating tau...');
for j=1:length(q)
    tau(j) = subs(Tau, 'k', q(j));
end

%%%%%%%%%%
% GRAPHICS %
%%%%%%%%%%
hold on;
h = loglog(q, tau(1:length(q)),'-');
set(h, 'Color', [delta*configuration delta*configuration
delta*configuration]);
drawnow
set(gca, 'xscale', 'log', 'yscale', 'log');
```