FinalCode_BowenCrossShoreEquations

Table of Contents

Cross-shore Sediment Transport using Shallow Water Wave Assumptions and Linear Airy Wave Theory Copyright (C) 2015 Alejandra C. Ortiz Developer can be contacted at aortiz88@alum.mit or aortiz4@ncsu.edu

License Information

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Initial Variables & Input Variables

```
clear all;
close all
clc;

%define unchanging variables
g = 9.81; %m/s2 gravity
es = 0.01; % suspended efficiency factor
es2 = 1; % EXTRA suspended efficiency suggested by Bailard 1981 in slope term - se
Cf = 0.01; % bed friction coefficient - suggestion from Bowen 1980 and Swenson 200
rho = 1.025; % water density
rhos = 2.65; % sediment density
porosity = 0.4;

%   Copyright 2015 Mirtech, Inc.
numpts = 5000;
%define initial profiles with x,z -- change number of points depending on
%resolution desired
x = linspace(4000,0,numpts);
z = linspace(50,2,numpts);
x = x(1:end-3);
z = z(1:end-3);

%Define Variables
T = 5:15; %varying wave period
H0 = 1:5; %varying deep-water wave height
```

```
%coresponds to v. fine, fine, medium, coarse and very coarse --
%p.197 Engelund & Hansen Mechanics of Sed Transport
ws = [.008 0.016 .033 .084 .16]; %varying fall velocities cm/s

lT = length(T);
lH = length(H0);
lz = length(z);
lw = length(ws);

z1 = repmat(z',[1 lT lH lw]);%just making giant matricies (4D) of
x1 = repmat(x',[1 lT lH lw]);%varying wave height, wave period
T1 = repmat(T,[lz 1 lH lw]);%and varying fall velocity

Ws = nan(size(T1));
for i = 1:lw
    Ws(:,:,:,i) = ws(i);
end

H01 = nan(size(T1));
for i = 1:lH
    H01(:,:,i,:) = H0(i);
end

z = z1;
x = x1;
T = T1;
H0 = H01;
ws = Ws;

%Define space for variables
L_s = nan(lz,lT,lH,lw);
L_sh = nan(lz,lT,lH,lw);
L_l = nan(lz,lT,lH,lw);
L_lh = nan(lz,lT,lH,lw);

%Unchanging
L0 = g.*T.^2./(2.*pi); %deep water wave length based on gravity & wave period
```

Case 1 -- using shallow water wave approximations

```
clc; close all

%Use Shallow Water Approxs waves
L_s = T.*sqrt(g.*z); %calculate local wave length with depth
H_s = H0.*sqrt(.5.*L0./L_s);  %calculate local wave height with depth

%sed transport
B = diff(z)./diff(x); %calculate slope
delz = abs(diff(z(1:2)));
%calculate eq. slope using shallow water wave
Bo_shallow = -3.*ws./(4.*es2.*sqrt(g.*z)) .* (5 + 3.*T.^2.*g./(4.*pi^2.*z));
```

```
%now calculate equilibrium shoreface distance (xeq)
xeq_s = zeros(size(T));
for l2 = 1:lw
    for k2 = 1:lH
        for j = 1:lT
            for i=length(z)-1:-1:1
                xeq_s(i,j,k2,l2) = xeq_s(i+1,j,k2,l2)-1/Bo_shallow(i,j,k2,l2).*del
            end
        end
    end
end
B(end+1,:,:,:) = NaN;%so same size as all other matrices

%the XShore sediment transport flux terms from Bowen
K = (es * Cf * rho * 16) / ((rhos-rho) * g * pi * 15); % [s2/m]

uo_s = (H_s.*sqrt(g))./(2.*sqrt(z)); %[m/s]
u1_s = (3.*H_s.^2.*sqrt(g))./(16.*z.^(3/2)); %[m/s]
u2_s = (3.*H_s.^2.*g.^(3/2).*T.^2)./(64.*pi.^2.*z.^(5/2)); %[m/s]

%this is cross-shore sediment flux equations
qs_s = -K.*(uo_s.^3)./ws.*[5.*u1_s + 3.*u2_s + (B.*es2./ws).*uo_s.^2]; %[m2/s]

%eq. slope
qeq_s = -K.*(uo_s.^3)./ws.*[5.*u1_s + 3.*u2_s + (Bo_shallow.*es2./ws).*uo_s.^2]; %

%Derivative dq/dx -- using Exner Equation -- these are the terms for
%Advection Diffusion Eq
uop_s = -(H_s.*sqrt(g))./(4.*z.^(3/2)); %[1/s]
u1p_s = -(9.*H_s.^2.*sqrt(g))./(32.*z.^(5/2)); %[1/s]
u2p_s = -(15*H_s.^2.*g.^(3/2).*T.^2)./(128.*pi.^2.*z.^(7/2)); %[1/s]

%This is the advection coefficient (Vc in paper)
V_sc = 5.*u1p_s.*uo_s + 15.*u1_s.*uop_s + ... %u1 terms
    3.*u2p_s.*uo_s + 9.*u2_s.*uop_s + ... %u2 terms
    (5.*Bo_shallow.*es2./ws).*uo_s.^2.*uop_s; %uo terms %[m/s2]
D_sc = es2.*uo_s.^3./ws; %[m2/s2] Diffusivity coefficient (Dc in paper)

Kpp_s = -K.*(uo_s.^2)./ws;%derivative of coefficient K

V_s = V_sc.*Kpp_s; %[m/s]!!! %advection term
D_s = D_sc.*Kpp_s; %[m2/s]!!! %diffusion term

Pe_s = V_s./D_s .* xeq_s; %Peclet
Tad_s = xeq_s./V_s; %timescale of advection (kinematic celerity)
Tdiff_s = xeq_s.^2./(-D_s);%Timescale of diffusivity
```

Case 4 -- using linear Airy wave theory

```
%for wave terms instead of shallow water wave assumptions
clc; close all
```

```
%deep water wave equations
c0 = L0./T; %deep water wave celerity
n0 = .5;
Cg0 = n0.*c0;%deep water group speed

%Use Full Linear Theory
L_lh = L0.*sqrt(tanh(2*pi.*z./L0)); %local wave length
k = 2.*pi./L_lh; %wave number
n = 0.5.*(1 + 2.*k.*z./sinh(2.*k.*z));
c = L_lh./T; %wave celerity
Cg = n.*c; %group speed
H_l = H0.*sqrt(Cg0./Cg); %local wave height

%sed transport
%Equilibrium slope
Bo_linear = -.75.*ws.*T./(L_lh.*es2) .* (5 + 3.*(csch(k.*z)).^2 );
delz = abs(diff(z(1:2)));

%sed transport
xeq_l = zeros(size(T));
for l2 = 1:lw
    for k2 = 1:lH
        for j = 1:lT
            for i=length(z)-1:-1:1
                xeq_l(i,j,k2,l2) = xeq_l(i+1,j,k2,l2)-1./Bo_linear(i,j,k2,l2).*del
            end
        end
    end
end

%Wave terms in Qs eq
uo_lh = (H_l.*pi)./(T.*sinh(k.*z)); %[m/s]
u1_lh = (3.*H_l.^2.*pi.^2)./(4.*T.*L_lh.*(sinh(k.*z)).^2); %[m/s]
u2_lh = (3.*H_l.^2.*pi.^2)./(4.*T.*L_lh.*(sinh(k.*z)).^4); %[m/s]

%Cross-shore sediment flux
qs_lh = -K.*(uo_lh.^3)./ws.*[5.*u1_lh + 3.*u2_lh + (B.*es2./ws).*uo_lh.^2]; %[m2/s
Kp_lh = -K.*(uo_lh.^3)./ws;

%equilibrium qs
qe_lh = -K.*(uo_lh.^3)./ws.*[5.*u1_lh + 3.*u2_lh + (Bo_linear.*es2./ws).*uo_lh.^2]

%Derivative dq/dx -- using Exner Equation
Lp_l = pi*(sech(2*pi.*z./L0).^2)./sqrt(tanh(2.*pi.*z./L0)); %[]

Hp_lh = -H0.*L0.*[Lp_l + 8*pi*csch(2.*k.*z) - ...
    32.*pi^2.*z./L_lh .*(1 - z.*Lp_l./L_lh).*coth(2.*z.*k).*csch(2.*z.*k)]...
    ./[2.*(8.*pi.*z.*csch(2.*k.*z) + L_lh).^2 .* ...
    sqrt(L0./(8*pi.*z.*csch(2.*k.*z) + L_lh))]; %

uop_lh = -(pi.*csch(k.*z))./(T.*L_lh.^2).*...
    [-L_lh.^2.*Hp_lh - 2.*pi.*z.*H_l.*Lp_l.*coth(k.*z) + ...
    2.*pi.*H_l.*L_lh.*coth(k.*z)]; %[1/s]
```

```
u1p_lh = -(3.*pi.^2.*H_l.*(csch(k.*z)).^2)./(4.*T.*L_lh.^3).*...
    [-2.*L_lh.^2.*Hp_lh + H_l.*L_lh.*Lp_l - 4.*pi.*z.*H_l.*Lp_l.*coth(k.*z) + ...
    4.*pi.*H_l.*L_lh.*coth(k.*z)]; %[1/s]

u2p_lh = -(3.*pi.^2.*H_l.*(csch(k.*z)).^4)./(4.*T.*L_lh.^3).*...
    [-2.*L_lh.^2.*Hp_lh + H_l.*L_lh.*Lp_l - 8.*pi.*z.*H_l.*Lp_l.*coth(k.*z) + ...
    8.*pi.*H_l.*L_lh.*coth(k.*z)]; %[1/s]

%Advection coefficient
V_lhc = 5.*u1p_lh.*uo_lh + 15.*u1_lh.*uop_lh + ... %u1 terms
    3.*u2p_lh.*uo_lh + 9.*u2_lh.*uop_lh + ... %u2 terms
    (5.*Bo_linear.*es2./ws).*uo_lh.^2.*uop_lh; %uo terms %[m/s2] SHOULD BE [m/s]

%Diffusion Coefficient
D_lhc = es2.*uo_lh.^3./ws; %[m2/s2]

Kpp_lh = -K.*(uo_lh.^2)./ws;%Derivative of K

V_lh = V_lhc.*Kpp_lh; %[m/s]!!! %Kinematic Celerity/Advection Term
D_lh = D_lhc.*Kpp_lh; %[m2/s]!!! % Diffusion Term

Pe_lh = V_lh./D_lh .* xeq_l;%Morphodynamic Peclet Number

Tad_lh = xeq_l./V_lh; %Timescale of Advection/kinematic celerity
Tdiff_lh = xeq_l.^2./(-D_lh); %Timescale of Diffusivity
```

Convert timescales to years & save data

```
TdiffYr_s = Tdiff_s./(3600*24*365);
TdiffYr_lh = Tdiff_lh./(3600*24*365);

TadYr_s = Tad_s./(3600*24*365);
TadYr_lh = Tad_lh./(3600*24*365);

%save ComputedData
```

Published with MATLAB® R2014b