```
function z1=detach_ftfs (z_init)
%
% detach_ftfs is a generic 1-D channel profile evolution solver.
% modified from bedrock_ftfs to allow a steady state profile as an initial condition.
% detach_ftfs solves the equation: dz/dt = U - KA^mS^n over the domain Xcrit ² x ² L
% (the fluvial part of a longitudinal profile).  No accounting for sediment flux and its↙
role
% erosion and sedimentation on the bed is included.
% bcflag is used to specify various boundary conditions at the downstream end (x = L)
% bcflag = 1: z(L) = constant -- fixed at erosional baselevel
% bcflag = 2: z(L) is lowered at a constant rate of baselevel fall
% bcflag = 3: dz/dx(L) = dz/dx(L-dx) (constant exit slope)
% icflag is used to specify various initial conditions (z0)
% icflag = 1: z0[Xcrit:L] is a linear ramp of slope = islope
% icflag = 2: z0[Xcrit:L] is a plateau of elev = plat_z
% icflag = 3: z0[Xcrit:L] is an intial steady state profile (Ui, Ki)
% time step dt is determined from standard stability condition for a FTCS solution of a
% linear advection equation (n = 1, U = 0).  Condition may need to be adjusted for more
% general conditions.
%
% units: all variables MUST be in meters-kilograms-years
%
% reset graphics
%
    hold off
%
% Set Parameter values for the run
%
    L       = 30000;
    Xcrit   = 300;
    Xnodes  = 200;      % Number of nodes in simulation, increase for greater accuracy
    Timeyr  = 30000;    % Duration of run in years

    m_n     = 0.5;
    n       = 1;
    ka      = 6.69;     % Hack's law
    h       = 1.67;     % Hack's law

    U       = 0.001;
    Ui      = 0.001;    % sets initial condition for icflag = 3
    K       = 0.00005;
    Ki      = 0.00005;  % sets initial condition for icflag = 3

    bcflag  = 1;
    icflag  = 3;
    islope  = 0.01;     % initial ramp slope for icflag = 1
    Blower  = 0.000;    % baselevel lowering rate for bcflag = 2
    plat_z  = 400;

    stabil  = 1;
%
% using revised criterion (remove ^n on dx), stabil = 10 needed for
```

```
% waterfall retreat problem *an extreme case*.
% Old problem of n>1 response to kp retreat
% was simply a stability problem.
% Determine values of derived constants. Determine dt for standard stability condition
%
    dx = L/Xnodes;
    m = m_n*n;
    dt = (dx)/(K*(ka^m)*(L^(h*m)))/stabil
%    dt = 0.5.*dx/(.5).^(n-1)/stabil
    nsteps = round(Timeyr/dt);
    nplot = 4                        % number of timesteps to plot
    tplot = round(nsteps/nplot);
%
% Set the x[Xcrit:L] array, adjust Xnodes so x[1:Xnodes] and initialize the z[Xcrit:L]↙
array
%
    x = [round(Xcrit/dx)*dx:dx:L];
    Xnodes = Xnodes - round(Xcrit/dx) + 1;
    z0 = zeros(size(x));
%
    if icflag==1
        z0 = (L + (-1.*x)).*islope;
%         for i = 1:Xnodes-7
%       z0(i) = z0(i) + 400;
%         end
%         z0(Xnodes-6)=300;
%         z0(Xnodes-5)=200;
%         z0(Xnodes-4)=50;
%         z0(Xnodes-3)=20;
%       z0(Xnodes) = 0;
    elseif icflag==2
        z0 = ones(size(x)).*plat_z;
        z0(Xnodes) = 0;
    elseif icflag==3
        x_temp = x.^(1 - (h.*m_n));
        z_coef = (Ui./Ki).^(1/n).*ka.^(-m_n).*(1 - h.*m_n).^-1;
        z0 = ((-1.*x_temp) + L.^(1 - h.*m_n)).*z_coef;
%       z0 = z0 + 200;
%       z0(Xnodes) = 0;
    elseif icflag==4
        z0 = z_init;
        z0 = z0 + 400;
        z0(Xnodes) = 0;
    end
    z1 = z0;
%
% Compute constants in the equation to be solved outside of the
% time loop to maximize efficiency.
%
    dU = U*dt;
    Kx = (K*(ka.^m)*dt/((dx).^n)).*(x.^(h*m));
    area = ka.*x.^h;
```

```
    for i = 1:Xnodes-1
        logslope(i) = log10((z0(i) - z0(i+1))/dx);
        logarea(i) = log10((area(i) + area(i+1))/2);
    end
%
% Plot initial condition in yellow
%
    figure(1)
    subplot(3,1,1)
    hold off
    plot(x,z0,'b')
    hold on
    xlabel 'distance (m)'
    ylabel 'elevation (m)'
    subplot(3,1,2)
    hold off
    plot(logarea, logslope, 'b')
    hold on
    xlabel 'log drainage area (m^2)'
    ylabel 'log channel gradient'
%
% Begin time loop, run from 1:nsteps
%
    for t = 1:nsteps
%
% loop through 1:Xnodes -1 for each time step
%
        for i = 1:Xnodes-1
            z1(i) = z0(i) + dU - Kx(i).*(abs(z0(i+1) - z0(i)))^n;
            z1(i) = max(z1(i),z0(i+1));
        end
%
% calculate last point according to boundary condition (bcflag)
%
        if bcflag==1
            z1(Xnodes) = z0(Xnodes);
        elseif bcflag==2
            z1(Xnodes) = z0(Xnodes) - (Blower.*dt);
        elseif bcflag==3
            z1(Xnodes) = z1(Xnodes-1) - (z0(Xnodes-1) - z0(Xnodes));
        end
%
% update z0,z1 for the next time loop and plot z2 if appropriate timestep (green)
%
        z0 = z1;
        if rem(t,tplot)==0
            for i = 1:Xnodes-1
                logslope(i) = log10((z1(i) - z1(i+1))/dx);
            end
            subplot(3,1,1)
            plot(x,z1,'g')
            subplot(3,1,2)
```

```
            plot(logarea,logslope,'g')
        end
%
% close time loop -- repeat for all t 1:nsteps
%
    end
%
% plot final solution in magenta and release hold on graphics
%
    for i = 1:Xnodes-1
        logslope(i) = log10((z1(i) - z1(i+1))/dx);
    end
    subplot(3,1,1)
    plot(x,z1,'m')
    subplot(3,1,2)
    plot(logarea,logslope,'m')
%   subplot(3,1,3)
%   plot(time,dzdt)
%   xlabel 'time (yr)'
%   ylabel 'dz/dt(Xc) (m/yr)'
    hold off


% Calculate the FINAL steady state solution for U and K.
    x_temp = x.^(1 - (h.*m_n));
    z_coef_ss = (U./K).^(1/n).*ka.^(-m_n).*(1 - h.*m_n).^-1;
    z0_ss = z1(Xnodes)+(((-1.*x_temp) + L.^(1 - h.*m_n)).*z_coef_ss);

    subplot(3,1,1)
    hold on
    plot(x,z0_ss,'k')
```