Appendix C: MATLAB Code Used for 1D Overpressure Modeling

This appendix provides detail on the Matlab code used to solve 1D overpressure. The code solves an equation that accounts for sedimentation and ice sheet loading:

$$K_z \frac{\partial^2 h}{\partial z^2} = S_s \left[\frac{\partial h}{\partial t} - \zeta \frac{\rho_i}{\rho_f} \frac{\partial \eta}{\partial t} - \zeta \frac{\rho_b - \rho_w}{\rho_f} \frac{\partial L}{\partial t} - P_{ext} \right], \qquad \text{(Equation C-1)}$$

where K_z is the hydraulic conductivity in the vertical direction, S_s is the specific storage, ρ_i, ρ_w, and ρ_b are the densities of ice, water, and fluid-saturated sediment, η and L are the elevations of the ice and sediment relative to sea level, respectively, and P_{ext} is an external source term which can be assigned to a particular layer to represent external fluid sources outside of the model domain (Dugan and Germaine, 2008). This groundwater flow equation (C-1) is solved with a fully implicit scheme which provides a solution that is unconditionally stable (Fletcher, 1997). The finite difference solution was compared to an analytical solution that describes the formation of a thick sedimentary package with a constant sedimentation rate overlying an impermeable base (Bredehoeft and Hanshaw, 1968; Gibson, 1958).

FD_MOD2.m

```
%------------------------------------------------------------------------%
%   1D pore pressure model for:
%   a) sediment loading and unloading
%   b) ice sheet loading and unloading
%   c) variable permeability
%
%  Subroutines and Functions used in program:
%  ( Fi_Iterate ) is uses to calculate pore pressure with each iteration
%  ( keffective ) calulates the effective permeability for the model domain
%  ( lith_D ) determines lithologic stress
%  ( ES_Matrix ) calculates the effective stress
```

```
%------------------------------------------------------------------------%

par_file        % par_file contains all input parameters

%--- Create Constant Vectors -------------------------------------------%
%------------------------------------------------------------------------%

depth=1:1:td;        % depth increment of model (1 meter)

hydro=depth*dw*9.8;  % hydrostatic pressure relative to top of model
                  % (current sealevel)

% matrix with pore pressure at each iteration
PoreTime=(zeros(1,d1-1),(d1:1:td))*(dw*9.8);

% matrix with lithologic stress at each iteration
LithTime=lith_D(d1, td, dw, ds);

%--- Pore Pressure For Time Steps ---------------------------------------%
%------------------------------------------------------------------------%

%--- Initial Pore Pressure at time 1:
%------------------------------------------------------------------------

poreT1=hydro(:,d1:td);    % pore pressure at T1, same as hydro, no loading
depthT1=depth(d1:td);     % depth of pore pressure

MEST1=ES_Matrix(PoreTime, LithTime);    % maximum effective stress for t1

%--- Pore pressure for time t2:
%------------------------------------------------------------------------

display ('ice sheet loading')

delt=(t1*365*24*60*60)/(d2-d1);     % size of time step per grid =
                              % total time / change in elevation

del_sed=-1;                       % sediment erosion per time step
del_ice=(hi-d1*(dw/di))/(d2-d1);    % ice growth per time step above min-
                              % flotation thickness

% ice must be more than flotation thickness
if hi < d1*(dw/di)
   del_ice=0;
   disp('Ice Less Than Floatation Thickness')
end

Bi=BL;      % B fore ice loading
Bs=BL;      % B for sediment loading

x=poreT1/(dw*9.8)-depthT1;        % initial pore pressure
x=(x,x(length(x)));              % base node for no flow boundary
ki=ones(1,length(x))*k1;         % vector of permiabilitys
```

```
% Create confining layer:
if clt == 0
   disp('No Confining Layer')
else
   for i = (d2-d1)+dcl:1:(d2-d1)+clt+dcl
      ki(i)=clp;  % node has permiability set by "clp"
   end
end

% loop for iterations: decrease grid size from erosion & load ice sheet
for n = 1:1:(d2-d1)
   HI=x(:,2:length(x));    % remove grid cell due to erosion
   ki=ki(:,2:length(ki));  % remove grid cell due to erosion
   ke=keffective(ki, 1);   % calculate new k effective

   % adjust boundary conditions
   HI(1)=(d1*(dw/di)+n*del_ice)*(di/dw)-(d1+n);
   %        (equivilent head of ice)  -(depth)

   if hi < d1*(dw/di)
      HI(1)=0;        % Head of seafloor if no ice is pressent:
   end

   FI_Iterate

   N=(d2-d1);
   % fill pore pressure matrix:
   file1=((zeros(1,d1-1+n),x(1,1:length(x)-1)+depth(d1+n:td)))*(dw*9.8);
   PoreTime=(PoreTime; file1);

   % fill lithology matrix:
   file1=lith_D(d1+n, td, dw, ds);
   file1=file1+del_ice*n*9.8*di;
   LithTime=(LithTime; file1);

end
x=x(1,1:length(x)-1);   % remove base boundary
ki=ki(1,1:length(ki)-1);
depthT2=depth(d2:td);
x=x+depthT2;           % add hydrostatic head
poreT2=x*(dw*9.8);      % pore pressure in (Pa)

% set eroded seds pore pressure to 0
for l = 1:1:length(PoreTime(:,1))
   for i = d1:1:d2-1
      PoreTime(l,i)=0;
   end
end

MEST2=ES_Matrix(PoreTime, LithTime);    % maximum effective stress for t2

%--- Pore presure at time t3:
```

```
%---------------------------------------------------------------------------
%
% *this is done in two stages:
% A) unoad ice sheet for time (t2)
% B) load glacigenic sediments for time (t2)
%

display ('ice sheet unloading')

Bi=BU;      % reduced B for deglaciation
Bs=BL;

%--- (A) -------------------------

delt=(t2*365*24*60*60);     % size of time step per grid = total time

del_sed=0;
del_ice=-(hi-d2*(dw/di));   %decline in ice to floatation thickness


HI=poreT2/(dw*9.8)-depthT2;
HI(1)=0;                  % top node has head of seafloor
HI=(HI,HI(length(HI)));     % base node for no flow boundary

ki=(ki,ki(length(ki)));     % initial k from previous time step
ke=keffective(ki, 1);

FI_Iterate

x=x(1,1:length(x)-1);        % remove extra base node
ki=ki(1,1:length(ki)-1);
x=x+depthT2;
x=x*(dw*9.8);

% fill pore pressure matrix:
file1=(zeros(1,d2-1),x)*(dw*9.8);
PoreTime=(PoreTime; file1);

% fill lithology matrix:
file1=lith_D(d2, td, dw, ds);
LithTime=(LithTime; file1);

poreT3A=x;

%--- (B) --------------------------

Bs=BL;

delt=(t2*365*24*60*60)/(d2-d3);     % size of time step per grid =
                              % total time / change in elevation
del_sed=1;
del_ice=0;
```

```
x=poreT3A/(dw*9.8)-depthT2;              % initial pore pressure
x=(x,x(length(x)));              % base node for no flow boundary

ki=(ki,ki(length(ki)));     % initial k from previous time step

for n = 1:1:(d2-d3)
   HI=(0,x);

   % adjust boundary conditions:
   HI(1)=0;           % new node has head of sea foor
   ki=(k2, ki);       % new node has perm of k2

   ke=keffective(ki, 1);

   FI_Iterate

   % fill pore pressure matrix:
   file1=((zeros(1,d2-1-n),x(1,1:length(x)-1)+depth(d2-n:td)))*(dw*9.8);
   PoreTime=(PoreTime; file1);

   % fill lithology matrix:
   file1=lith_D(d2-n, td, dw, ds);
   LithTime=(LithTime; file1);

end
depthT3=depth(:,d3:td);

x=x(1,1:length(x)-1);       % remove extra base node
ki=ki(1,1:length(ki)-1);
x=x+depthT3;
x=x*(dw*9.8);
poreT3=x;

MEST3=ES_Matrix(PoreTime, LithTime);    % maximum effective stress for t3

%--- Pore pressure at time t4:
%-------------------------------------------------------------------------

display ('Pleistocene sedimentation')

Bs=BL;

delt=(t3*365*24*60*60)/(d3-d4);
del_sed=1;
del_ice=0;

x=poreT3/(dw*9.8)-depthT3;        % initial pore pressure
x=(x,x(length(x)));              % base node for no flow boundary

ki=(ki,ki(length(ki)));     % initial k from previous time step

for n = 1:1:(d3-d4)
```

```
    HI=(0,x);

    % adjust boundary conditions:
    HI(1)=0;  % new node has head of sea foor

    % determine new effective stress:
    ki=(k3, ki); % new node has perm of k3
    ke=keffective(ki, 1);

    %artificial layer loading
    if n >= ((d3-d4)-3)
       for i=(d2-d3)+1+n+do:1:(d2-d3)+1+n+do+lo
          HI(i)=HI(i)+mo;
       end
    end

    FI_Iterate

    % fill pore pressure matrix:
    file1=((zeros(1,d3-1-n),x(1,1:length(x)-1)+depth(d3-n:td)))*(dw*9.8);
    PoreTime=(PoreTime; file1);

    % fill lithology matrix:
    file1=lith_D(d3-n, td, dw, ds);
    LithTime=(LithTime; file1);

end
x=x(1,1:length(x)-1);        % remove extra base node
ki=ki(1,1:length(ki)-1);
depthT4=depth(:,d4:td);
x=x+depthT4;
x=x*(dw*9.8);
poreT4=x;

MEST4=ES_Matrix(PoreTime, LithTime);    % maximum effective stress for t4

%--- Lithologic Pressure For Time Steps ---------------------------------%
%-----------------------------------------------------------------------%

lithT1=(depthT1-d1)*ds*9.8+d1*dw*9.8;

lithT2=(depthT2-d2)*ds*9.8+hi*di*9.8;   % note, density of ice is used

if hi < d1*(dw/di)
     lithT2=(depthT2-d2)*ds*9.8+d2*dw*9.8;
end

lithT3=(depthT3-d3)*ds*9.8+d3*dw*9.8;

lithT4=(depthT4-d4)*ds*9.8+d4*dw*9.8;

%--- Effective Stress For Time Steps ------------------------------------%
%-----------------------------------------------------------------------%
```

```
esT1=lithT1-poreT1;

esT2=lithT2-poreT2;

esT3=lithT3-poreT3;

esT4=lithT4-poreT4;

%--- Plot Results -------------------------------------------------------%
%-----------------------------------------------------------------------%

figure(1)   % pore pressure predicted from model

subplot(2,4,1)
pore=poreT1;
depthP=depthT1;
lith=lithT1;
es=esT1;
max_ES=MEST1;
sf=d1;
S_plot1
title('initial condition')
subplot(2,4,5)
S_plot2

subplot(2,4,2)
pore=poreT2;
depthP=depthT2;
lith=lithT2;
es=esT2;
max_ES=MEST2;
sf=d2;
S_plot1
title('erosion & ice loading')
plot((-10000000,12000000),(d2, d2),'r--')
subplot(2,4,6)
S_plot2
plot((-10000000,12000000),(d2, d2),'r--')

subplot(2,4,3)
pore=poreT3;
depthP=depthT3;
lith=lithT3;
es=esT3;
max_ES=MEST3;
sf=d3;
S_plot1
title('ice retreat & sedimentation')
plot((-1000000,12000000),(d2, d2),'r')
subplot(2,4,7)
S_plot2
plot((-1000000,12000000),(d2, d2),'r')
```

```
subplot(2,4,4)
pore=poreT4;
depthP=depthT4;
lith=lithT4;
es=esT4;
max_ES=MEST4;
sf=d4;
S_plot1
title('pressent')
plot((-1000000,12000000),(d3, d3),'k')
plot((-1000000,12000000),(d2, d2),'r')
subplot(2,4,8)
S_plot2
plot((-1000000,12000000),(d3, d3),'k')
plot((-1000000,12000000),(d2, d2),'r')


% Deviatoric Stress

%figure(3)
%dev=diff(poreT3-depthT3*dw*9.8);
%depth_dev=depthT3(:,2:length(depthT3))-.5;
%plot(dev,depth_dev)
%set(gca,'YDir','reverse')
%axis((-10000,10000,0,600))
%hold on


figure(2) % changes in effective stress predicted from model

%--- Change in Effective Stress from T3 to T4
%subplot (1,3,1)
%del_es=esT4(:,(d3-d4)+1:length(esT4))-esT3; % change in effective stress
%plot(del_es,depthT3,'k','linewidth',2)
%set(gca,'YDir','reverse')
%axis((0,3000000,0,600))
%itle('Current - Previous (deglaciation)')
%hold on
%plot((0,12000000),(d3, d3),'k')
%plot((0,12000000),(d2, d2),'r')

subplot (1,3,1)
del_st=MEST4(:,d4:td)-esT4; % difference in max effective stress and present
plot(del_st,depthT4,'k','linewidth',2)
set(gca,'YDir','reverse')
axis((0,1000000,0,600))
title('Max ES - Present ES')
hold on
plot((0,12000000),(d3, d3),'k')
plot((0,12000000),(d2, d2),'r')

subplot (1,3,2) % Predicted Porosity
```

```
Ph0=0.4;              % initial porosity
BC=5E-8;              % Bulk Compressibility
mes=MEST4(1,d4:td);   % Maximum Effective Stress

Ph=Ph0*exp(-BC*mes);

plot(Ph,depthT4,'k','linewidth',2)
set(gca,'YDir','reverse')
axis((0.25,.5,0,600))
title('Predicted Porosity')
xlabel('porosity')

% Han's Empirical relationship

C=.4;
%Pe=(lithT4-poreT4)*.01;

%Pe=(lithT4-poreT4)*(1/100000000);
%Vp=5.77-6.94*Ph-1.73*sqrt(C)+0.446*(Pe-1.0*exp(-16.7*Pe))

%Tosaya's empiricle relationship for shaley sandstones
Vp=5.8-8.6*Ph-2.4*C;

subplot (1,3,3)
plot(Vp,depthT4,'k','linewidth',2)
hold on
set(gca,'YDir','reverse')
axis((1.0,2.5,0,600))
title('Predicted Velocity')
xlabel('velocity (km/s)')
```

FI_Iterate.m

```
% FI_iterate
%
% model input = HI (head initial (m) )
% model output = x (head out (m) )

%------------------------------------------------------------------------

delx=1;

%--- Calculate K Effective (ke) -----------------------------------------%
%-----------------------------------------------------------------------%

Tx=((ke*dw*9.8)/(vis));
S=Ss;

%Tx=Tx*0+K; %Note, special case just for benchmark
```

```
% constant used in numerical approixmation:

alpha=(delt/(delx^2))*(Tx/S);        % Fluid transport term
beta=Bi*(di/dw)*del_ice;             % Ice loading term
gama=Bs*((ds-dw)/dw)*del_sed;        % Sed loading term

% make d matrix
d=(HI(2)+alpha(1)*HI(1)+beta+gama);
for i = 3:length(HI)-2
    d(i-1)=HI(i)+beta+gama;
end
d(length(HI)-2)=HI(length(HI)-1)+beta+gama;

% make values for compressed tridiagonal matrix
a=-alpha';
b=(1+2*alpha)';
b(length(b))=(1+alpha(length(alpha)));
c=-alpha';

x=TDMAsolver(a, b, c, d);

%fill in ends of X wich are fix as the boundary condition

file1=();
for i = 1:length(x)
    file1(i+1)=x(i);
end
file1(1)=HI(1);
file1(length(HI))=x(length(x));
```

keffective.m

```
function ( ke ) = keffective ( ki, delx )

% Outpuy ( ke ) is tje effective permiability based on the array with
% intrinsic permiability at each node (ki)
% ( ke ) will have two less nodes than ( ki )

ke=();
for i = 2:1:length(ki)-1

    %ke(i)=((3*delx)/((delx/ki(i-1))+(delx/ki(i))+(delx/ki(i+1))));
    ke(i)=((2*delx)/((delx/(2*ki(i-1)))+(delx/ki(i))+(delx/(2*ki(i+1)))));

end
ke=ke(:,2:length(ke));

end
```