**Mefisto V2 User’s Manual**

John N. Hooker (jzh497@psu.edu)

With contributions from Don Fisher & David Oakley

MEFISTO: the Mineralization, Earthquake, and Fluid-flow Integrated SimulaTOr is a Matlab script written using Matlab v. 2019a. Paste the orange text at the end of this manual into a Matlab script to run.

MEFISTO extends the model used in Fisher et al. (2019), which was based upon a numerical representation of a 2D grid of sliding blocks created by Huang et al. (1992). A rigid subducting plate moves at constant velocity *r* with respect to a rigid overriding plate. The subducting plate is connected to a grid of sliding blocks via leaf springs, which have a bending moment. One such leaf spring connects each block to the subducting plate; each block is connected via springs to its nearest neighbors in the X (strike-parallel) direction and the Y (dip-parallel and transport-parallel) direction, creating a grid. X-parallel springs are leaf springs; Y-parallel springs are coil springs, which have a linear elastic response to stretch. Each cell makes a square footprint upon the interface and has a predetermined height. The blocks are all held against the overriding plate in frictional contact. Cell motion in X is prohibited.

The magnitude of the shear stress, *SS*, resolved upon the interface is

determined by the positions of its neighboring cells, to which is is connected via connecting springs (Huang et al., 1992):

SSij = Yij + α (4Yij - Yij+1 - Yij-1 - Yi+1j - Yi-1j) (1)

where Yij represents the distance along Y between a cell’s current location and its at-rest position and α is the ratio between the spring constants of coupling and leading springs--i.e., springs linking cells to one another, and springs linking each cell to the subducting plate. The normal stress, SN, resolved on the interface is

SNij = ρ g hij – Pij + SSij /sin(2d)× [cos(2d)-1](2)

where *d* is the dip of the interface and Pij is the fluid pressure (see below).

Sliding occurs once *S* exceeds a Mohr-Coulomb failure criterion:

SSij > C + μ SNij’ (3)

Where C is the cohesion and *μ* is the coefficient of internal friction, here using a default value of 0.1 (Byrne and Fisher, 1990).

The cell slides forward by a slip distance, *d*, which is that of a simple harmonic oscillator:

d = (-2/5)(S-1/φ) (4)

which becomes again locked in friction once its velocity reaches zero (Huang et al., 1992). φ is the ratio of static to dynamic friction. A sliding block changes the stresses upon its neighboring blocks by virtue of their coupling springs, so that in many cases those neighboring blocks are induced to sliding. A rupture ceases once no cells are stressed past their critical failure stress.

Mineral precipitation sites are nucleated at model cells with a probability over time based on temperature, consistent with quartz precipitation being is limited by kinetics (Lander et al., 2008). Nucleation therefore follows an Arrhenius law:

p = exp(-E/RT) (5)

where *R* is the gas constant (8.314 × 10-3 kJ/mol K), *T* is temperature in Kelvins, and *E* is an activation energy, which can vary depending on the mineral being precipitated. Once nucleated, a mineralized cell’s cohesion increases with the log of time, up to a maximum:

ΔC = min(log[t] × Astr exp[-G/RT], ΔCmax) (6)

where *t* is the time in seconds since the time of asperity nucleation, *Astr* is a pre-exponential strengthening constant with units of stress, and *G* is the activation energy for strengthening.

Each cell represents a porous medium (Hooker and Fisher, submitted) whose fluid content undergoes mass-conserving Darcian flow with its neighboring cells over time. The model is run in time increments. After each time increment each cell is checked against its failure criterion; the time increment size is sufficiently small so that during most increments no cells fail. It is possible that multiple, connected or unconnected, cells reach their failure stress during the same increment.

Fluid flow in the model is based on conservation of mass. Each cell is initialized having hydrostatic fluid pressure and a default volume, porosity, and permeability. Brine density is held constant at 1100 kg m-3. Fluid motion arises from pressure changes due to two sources: (1) fluid production, which proceeds by adding fluid mass within any cell at a predefined, constant rate, and (2) porosity changes. Porosity is reduced by mineral precipitation and restored to its default value during earthquake ruptures.

Fluid production is constant over time and assigned at each cell. By default fluid production rate is constant along strike and follows a gaussian distribution parallel to dip that has a peak value at 0.9*nj* and a standard deviation of two cell lengths. The magnitude of fluid production can be varied using a factor, F, having units kg s-1 m-2, such that the fluid generated in any cell per unit time is:

fcell = [1/ √(2 π σ2)] exp[(j – 0.9nj)2/2 σ2] l h F dt (7)

where *σ* is the standard deviation of the fluid production profile, *j* is the cell’s *j*- (dip-parallel) coordinate, and *l* and *h* are cell length (strike-parallel) and height, respectively.

The non-hydrostatic component of fluid pressure is calculated as:

P = min[(n\*/n × m/m\* – 1)/ β, (ρ g h) + ΔC St] (8)

where *n* is porosity, *m* is the mass of fluid in a cell, *n\** and *m\** are the default porosity and fluid mass, respectively, for model cells having hydrostatic pressure (e.g., Ghani et al., 2013), and *β* is water compressibility (here, 4.6 × 10-4 MPa-1). Default mass is based on a fluid density of 1100 kg m-3. P is limited to a maximum value, *(ρ g h) +* *ΔC St*, corresponding to the overburden stress plus the tensile strength of the rock. *St* is defined along (0,1]and is the factor by which the tensile strength is lower than the cohesion. The overburden stress (*ρ g h*) is calculated as the weight of the water-saturated, porous rock column overlying each cell. The height *h* varies laterally as a function of the interface dip; *g* is 9.8 m s-2; density *ρ* is based on the porosity volume-fractions of a solid component (2700 kg m-3) and fluid component (1100 kg m-3).

Cementation can reduce pore space or permeability (*K*), or both, by a user-defined factor for each. Reduction of both pore space and permeability scales linearly with the amount by which the cell has been strengthened by cementation:

n = n\* - nR n\* ΔC/ΔCmax (9a)

K = K\* - KR K\* ΔC/ΔCmax (9b).

*K\** is a cell’s default permeability. *nR* and *KR* are defined on [0,1) and represent the factor by which porosity and permeability, respectively, are reduced upon complete cementation of the cell.

Fluid motion obeys Darcian flow, preserving fluid mass, so that at each time increment the change in fluid mass is determined by the non-hydrostatic pressure difference and the harmonic mean of the neighboring permeabilities:

dmij/dt = 4 × h × [(Pi+1j – Pij) × 2/(1/Ki+1j + 1/Kij) + (Pi-1j – Pij) × 2/(1/Ki-1j + 1/Kij) + (Pij+1 – Pij) × 2/(1/Kij+1 + 1/Kij) + (Pij-1 – Pij) × 2/(1/Kij-1 + 1/Kij)] + fcell (10)

and *K* in the model has units of time; *K* can be converted to hydraulic conductivity (m s-1) by multiplying by the acceleration of gravity.

The sequence for each iteration is as follows. First, cement is precipitated, producing changes in strength, pore space, and permeability. Second, fluid mass changes in response to both pressure-driven flow and fluid generation. Third, fluid pressure is recalculated according to the updated fluid mass and pore space (Equation 9). Finally, the subducting plate moves another increment and each cell is checked for failure.

Failure results in cells sliding forward (down-dip) a distance determined by the stress imparted by neighboring spring-loaded cells. Also upon failure, strength, porosity, and permeability return to their default, uncemented state. Following this, all cells are again checked for failure, and sliding and concomitant cement un-bonding continue until no cells fail. These changes are regarded as instantaneous. Once the rupture ceases, the time-dependent processes of cement precipitation and fluid flow proceed again.

Model hydraulic boundary conditions are as follows. The down-dip and lateral edges are no-flow boundaries, as is the interface-parallel boundary on the footwall side. The hanging wall boundary acts as a valve. Once the pressure reaches its maximum (Equation 8), fluid is ejected out the overriding plate, through hypothetical opening-mode fractures, and lost. The updip limit of the interface is a constant pressure boundary, held to hydrostatic and assumed to represent the intersection of the subduction interface with the sea floor.

The interface is assigned physical dimensions, dip angle, and temperature.

The model initializes with each cell’s Y coordinate assigned a random number, and each cell having the default porosity and fluid mass. During an initial period the model achieves a steady state with respect to seismic rupture behavior and fluid flow. A “burn-in” period consists of a user-defined number of iterations to ignore at the beginning, while the model achieves steady-state.

The model should be run with sufficiently small time increments to ensure that no cell runs out of fluid. The routine terminates if any cell’s fluid mass reaches zero.

The code is below. It begins with an explanation of user-defined variables, below which they may be set within lines ending in the %%%User%%% designation.

References

Byrne and Fisher, 1990, Evidence for a weak and overpressured décollement beneath sediment-dominated accretionary prisms: Journal of Geophysical Research, v. 98, no. B6, p. 9081-9097.

Fisher, D.M., Hooker, J.N., and Oakley, D.O.S., 2019, Numerical models for slip on the subduction interface motivated by field observations: Lithosphere, v. 11, no. 3., p. 322-332, doi:10.1130/L1008.1.

Ghani, I., Koehn, D., Toussaint, R., and Passchier, C.W., 2013, Dynamic development of hydrofracture: Pure and Applied Geophysics, v. 170, p. 1685-1703. doi:10.1007/s00024-012-0637-7.

Hooker and Fisher, submitted, How cementation and fluid flow influence slip behavior at the subduction interface: In review, Geology.

Huang, J., Narkounskaia, G., and Turcotte, D.L., 1992, A cellular-automata, slider-block model for earthquakes II, Demonstration of self-organized criticality for a 2-D system: Geophysical Journal International, v. 11, p. 259-269, doi:10.1111/j.1365-246X.1992.tb00575.x.

Lander, R.H., Larese, R.E., and Bonnell, L.M., 2008, Toward more accurate quartz cement models: The importance of euhedral versus noneuhedral growth rates: AAPG Bulletin, v. 92, no. 11, 1537-1563, doi:10.1306/07160808037.

%Welcome to MEFISTO, the Mineralization, Earthquake, and Fluid-flow

%Integrated SimulaTOr. For more information, visit www.mefisto.org.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1. DEFINITIONS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1a. Basic geometry variables.

% block\_height: Cell height (m). Normal to interface.

% dip: Interface dip (radians).

% dt: time (s) per model increment.

% L: Dip-parallel length of interface (km).

% nx: Number of cells in the x (strike) direction.

% ny: Number of cells in the y (dip) direction.

% plate\_rate: Plate convergence rate (m/yr).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1b. Model execution variables.

% aftershocks: Number of ruptures that are mapped after a large event (see

%'rupture\_snap') occurs

% burn\_in: Number of increments to ignore at the beginning. Intended for

%model to reach steady-state conditions before generating output.

% frame\_int: Number of increments between movie frames.

% init\_conditions: ['file', 'rand', or 'flat'] Controls the initial distributions of cell locations and asperitized cells.

% init\_loc\_max = 1; If init\_conditions is 'rand', this is the maximum initial loc value.

% mapruptures: [1 or 0] Option of whether to save a map of aftershock

%ruptures.

% n\_incs = Number of time increments to run through.

% rupture\_snap: Earthquake magnitude above which subsequent ruptures [n =

%aftershocks] are mapped.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1c. Model physical variables.

%1c1. Elasticity

% D\_phi\_asp: Change in ratio of static to dynamic friction for asperities.

% kc = 3; %Spring constant of coupling springs, which connect neighboring cells.

% kl = 1; Spring constant of leading springs, which connect each cell to the subducting slab.

% max\_dstrength: Max shear strength added to asperities (MPa).

% non\_asp\_str: Failure shear stress of non-asperity cells (MPa).

% phi [>1]: Ratio of static to dynamic friction.

% tensile\_strength\_factor: ratio of max\_dstrength to tensile strength.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1c2. Plasticity

% flowlaw [1 or 0]: Option to incorporate temperature-dependent crystal-plastic flow.

% F: Pre-exponential term for flow law.

% Fe: Exponential term for flow law.

% Fx: Power-law exponent for flow law.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1c3. Hydraulics

% defperm: Default hydraulic conductivity of cells (m/s)

% minKfactor [0 1): factor by which a sealed cell's permeability is reduced (1 means reduced to zero)

% defpore = 0.01; default porosity for unhealed cells (unitless).

% minporefactor [0 1): factor by which a sealed cell's porosity is reduced

% water\_comp = 4.6e-4;%per megapascal. use 4.6e-4 for water

% sea\_floor [0 or 1]: option for sea-floor constant pressure boundary

% condition at top; otherwise top is no-flow boundary.

% sea\_floor\_pressure: sea floor pressure, MPa, for sea floor constant-pressure boundary condition

% fluidmean: Depth of max fluid production (units of cell rows).

% fluidsd: Standard deviation of fluid production zone. Controls height of fluid production zone, which is normally distributed parallel to dip and constant parallel to strike.

% fluid\_prod\_mag = 1e-10; %Controls amount of fluid production. units are kg/s/mwidthx)/m(height); scales with block volume but not porosity.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1c4. Temperature and mineral controls.

% A: Strengthening pre-exponential factor.

% C: Asperity nucleation factor.

% Ea: Activation energy for asperity nucleation.

% G: Activation energy for strengthening.

% R: Gas constant.

% Tmin: Temperature at the top of the slab (celsius).

% Tmax [>Tmin]: Temperature at the bottom of the slab (celsius).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all

cc = clock;

tic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Use this section if you wan to run MEFISTO multiple times and make a

%matrix of ouctomes

%Enter parameters to vary. Be sure to comment them out below if you want

%them to vary; otherwise they will be overwritten below.

%If you don't want anything to vary, then just enter loop sizes = 1 here

for O = 1:1

 for P = 1:1

 fluid\_prod\_mag = 10^-(O-1);

 defperm = 10^(P-11);

 fluid\_prod\_mag\_stats(O,P) = fluid\_prod\_mag;

 defperm\_stats(O,P) = defperm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Model parameters.Change variables followed by '%%%User%%%' as desired.

nx = 50; %%%User%%%

ny = 50; %%%User%%%

n\_incs = 2000000;%%%User%%%

burn\_in = 100000; %%%User%%%

init\_conditions = 'flat'; %%%User%%%

init\_loc\_max = 1; %%%User%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Simulation options

mapruptures = 1;%%%User%%%

frame\_int = 1000;%%%User%%%

rupture\_snap =9; %%%User%%%

aftershocks = 50;%%%User%%%

aftershock\_count = aftershocks;%%%User%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Spring material properties

phi = 1.5; %%%User%%%

D\_phi\_asp = 0; %%%User%%%

kc = 3; %%%User%%%

kl = 1; %%%User%%%

alpha = kc/kl;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Temperature and mineral parameters

non\_asp\_str = 1; %%%User%%%

max\_dstrength = 40; %%%User%%%

tensile\_strength\_factor = 0.1; %%%User%%%

Ea = 54; %%%User%%%

C = 1e-3; %%%User%%%

A = 1e7; %%%User%%%

G = 54; %%%User%%%

Tmin = 25; %%%User%%%

Tmax = 250; %%%User%%%

R = 8.314e-3;

T = repmat((Tmin:(Tmax-Tmin)/(ny-1):Tmax)',1,nx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Model geometry parameters

L = 130; %%%User%%%

dip = 10 \* pi/180; %%%User%%%

block\_height = 1; %%%User%%%

plate\_rate = 0.05; %%%User%%%

dt = 5000; %%%User%%%

plate\_rate\_persec = plate\_rate/365/24/3600;

rupture\_snap = 10^((rupture\_snap-2/3\*log10(32e16)+10.7)\*1.5);

block\_area = (L/ny)^2\*1e6;

time\_increment = dt;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Fluid parameters

defperm = 1e-8; %%%User%%%

defK = defperm/9.8; %%%User%%%

K = ones(ny,nx).\*defK;

minKfactor = 0.99; %%%User%%%

defpore = 0.01; %%%User%%%

porosity = defpore.\*ones(ny,nx);

minporefactor = 0; %%%User%%%

water\_comp = 4.6e-4;

sea\_floor = 1; %%%User%%%

sea\_floor\_pressure = 0; %%%User%%%

overburden = ones(ny,nx)\*9.8\*(2400-1100)/1000000\*sin(dip)\*L\*1000/ny.\*(1:ny)';

fluidmean = ny\*0.7; %%%User%%%

fluidsd = 3; %%%User%%%

fluid\_prod\_mag = 1e-10; %%%User%%%

defmass = defpore\*block\_area\*block\_height\*1100;

fluid\_mass = porosity.\*block\_area.\*block\_height.\*1100;%mass filling porosity initially, density 1.1 g/cc

fluid\_pressure = (defpore./porosity.\*fluid\_mass./defmass-1)./water\_comp;% + overburden\*(1.1/2.4);%nonhydrostatic component; hydrostatic component omitted and added to failure crit

fluid\_prod = repmat(1/(sqrt(2\*pi\*fluidsd^2)).\*exp(-((1:ny)-fluidmean).^2./(2\*fluidsd^2))',1,nx)\*sqrt(block\_area)\*block\_height\*fluid\_prod\_mag;%gaussian fluid production distribution with depth.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Flow law parameters

F = 1e1; %%%User%%%

Fe = 90; %%%User%%%

Fx = 1;

flowlaw = 1; %%%User%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch init\_conditions

 case 'file'

 loc = load('loc\_init.csv'); %Location of each slider block relative to rigid block.

 asp = logical(load('asp\_init.csv')); %Logical array of whether or not a cell is an asperity.

 case 'rand'

 loc = init\_loc\_max\*rand(ny,nx);

 asp = logical(round(rand(ny,nx)));

 case 'flat'

 loc = zeros(ny,nx);

 asp = false(ny,nx);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%placeholder variables

rupture\_slip = zeros(1,n\_incs); %Total slip in each rupture.

slip\_surplus = zeros(1,n\_incs); %Tracks the slip surplus (+) or deficit (-) at the start of each rupture.

nloops = 0; %Number of times through the while loop involved in each rupture.

rupture\_area = zeros(1,n\_incs); %Number of cells that slip in each rupture, not counting cells that slip more than once as extra cells.

rupture\_map = zeros(ny,nx); %map of cells that slip during any single rupture.

fail\_cells = false(ny,nx); %Initially no cells are failing; even if they are, the first run through the while loop will determine that and go from there.

cum\_slip = zeros(ny,nx); %Tracks the cumulative slip on each cell.

dstrength = zeros(ny,nx); %added strength of each cell, will be based on thermal exposure since last rupture.

aspage = zeros(ny,nx);%age of each asperity

loc\_frame = zeros(ny,nx,n\_incs/frame\_int+1-burn\_in); %Stores a frame of the cumulative slip every frame\_int ruptures.

dstrength\_frame = zeros(ny,nx,n\_incs/frame\_int+1-burn\_in); %Make a movie of cell strength

fluid\_mass\_frame = zeros(ny,nx,n\_incs/frame\_int+1-burn\_in);

fluid\_pressure\_frame = zeros(ny,nx,n\_incs/frame\_int+1-burn\_in);

rupturemap\_frame = zeros(ny,nx,1);

aftershock\_frame = zeros(ny,nx,1);

fluid\_loss\_frame = zeros(ny,nx,1);

fail = zeros(ny,nx);

rupture\_init\_X = zeros(1,n\_incs);

rupture\_init\_Y = zeros(1,n\_incs);

rupture\_time = zeros(1,n\_incs);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Begin simulation

for n = 1:n\_incs %Run through the total number of increments.

 time\_increment = dt\*(10^(-log10(defperm)-6))^(n<(burn\_in\*0.9));% This step forces the model to increase time-steps during the burn-in, so that equilibrium fluid conditions are reached.

 if min(min(fluid\_mass)) == 0

 disp('Error: A cell has run out of fluid.');

 break

 end

 slip\_surplus(n) = sum(sum(loc)); %Slip surplus (+) or deficit (-) at the beginning of the next rupture.

 if min(min(fail)) <= 0

 [a,b] = min(min(fail));

 [c,d] = min(min(fail,[],2));

 rupture\_init\_X(n) = b;

 rupture\_init\_Y(n) = d;

 rupture\_time(n) = time\_increment/365/24/3600\*(n-1);%yr units

 end

 while min(min(fail)) <= 0 %1e-15) %Keep running through the rupture until no cells are displaced past their point of failure.

 nloops = nloops+1;

 stress = loc+alpha\*([loc(:,1:nx-1)-loc(:,2:nx),zeros(ny,1)]...

 +[loc(1:ny-1,:)-loc(2:ny,:);zeros(1,nx)]...

 +[zeros(1,nx);loc(2:ny,:)-loc(1:ny-1,:)]...

 +[zeros(ny,1),loc(:,2:nx)-loc(:,1:nx-1)]);%Shear stress on each slider block

 fail = dstrength+non\_asp\_str+0.1\*(overburden-fluid\_pressure+stress/sin(2\*dip)\*(cos(2\*dip)-1))-stress;

 fail\_cells = fail<=1e-15; %Logical array telling which cells fail. Changed to 1e-15 to deal with rounding errors.

 slip = -2\*(stress-1./(phi+D\_phi\_asp.\*asp))/(1+4\*alpha); %Distance each block would slip if it were to fail.

 rupture\_map = or(rupture\_map>0,fail\_cells>0); % rupture map is a logical of each cell that has slipped in this rupture.

 loc(fail\_cells) = loc(fail\_cells)+slip(fail\_cells); %Move failing cells by their (negative) displacement to bring them to failure.

 asp(fail\_cells) = false; %Reset the cells that failed so they are no longer asperities.

 dstrength(fail\_cells) = 0; %Reset the added strength to failed cells back to zero.

 porosity(fail\_cells) = defpore; %Set the porosity of failed cells to the default, unhealed value.

 fluid\_pressure(fail\_cells) = min(overburden(fail\_cells)+dstrength(fail\_cells)\*tensile\_strength\_factor,(defpore./porosity(fail\_cells).\*fluid\_mass(fail\_cells)./defmass-1)./water\_comp); % drop pressure in failed cells according to porosity; no flow during propagation

 if sea\_floor == 1

 fluid\_pressure(1,:) = sea\_floor\_pressure;%keep top row at constant pressure

 fluid\_mass(1,:) = porosity(1,:).\*block\_area.\*block\_height.\*1100;

 end

 aspage(fail\_cells) = 0;%reset the time since asperitization of each failed cell to zero.

 rupture\_slip(n) = rupture\_slip(n)+sum(slip(fail\_cells)); %Add to the total amount of displacement during this rupture.

 cum\_slip = cum\_slip+fail\_cells;

 rupture\_area(n) = sum(sum(rupture\_map)); %Rupture area is the total of all cells that slipped in that rupture.

 if nloops>2\*nx\*ny

 disp('Error: Rupture not stopping')

 break

 end

 end

 if mapruptures == 1

 if n > burn\_in

 %max\_slip(n) = min(min(cum\_slip)); %Used to track max slipping cell per rupture; uncomment with cum\_slip as desired

 if rupture\_area(n) > 0

 if aftershock\_count < aftershocks

 aftershock\_frame(:,:,size(rupturemap\_frame,3)\*aftershocks+aftershock\_count) = rupture\_map.\*aftershock\_count;

 aftershock\_count = aftershock\_count + 1;

 end

 end

 if -rupture\_slip(n)\*block\_area>=rupture\_snap

 rupturemap\_frame(:,:,size(rupturemap\_frame,3)+1) = -cum\_slip;

 init\_Xf(size(rupturemap\_frame,3)) = b;

 init\_Yf(size(rupturemap\_frame,3)) = d;

 aftershock\_count = 1;

 end

 end

 end

 rupture\_map = zeros(ny,nx); %Clear the rupture map.

 cum\_slip = zeros(ny,nx);

 nloops = 0;

 rn = rand(size(asp)); %

 asp(~asp & rn<real(1-((1-exp(-Ea./(R\*(T+273.15)))).^(time\_increment\*C)))) = true; %Nucleate asperities.

 aspage = aspage + asp.\*time\_increment;

 dstrength = asp.\*min(max\_dstrength,log(aspage+1).\*(A.\*exp(-G./(R\*(T+273.15)))));

 K = defK - defK\*minKfactor.\*dstrength./max\_dstrength;

 porosity = defpore - defpore\*minporefactor\*(dstrength./max\_dstrength);

 d\_fluid\_mass = 4\*1000000\*block\_height\*-time\_increment.\*([fluid\_pressure(:,1:nx-1)-fluid\_pressure(:,2:nx),zeros(ny,1)]...

 .\*[(K(:,1:nx-1)+K(:,2:nx))./2,zeros(ny,1)]...

 +[fluid\_pressure(1:ny-1,:)-fluid\_pressure(2:ny,:);zeros(1,nx)]...

 .\*[(K(1:ny-1,:)+K(2:ny,:))./2;zeros(1,nx)]...

 +[zeros(1,nx);fluid\_pressure(2:ny,:)-fluid\_pressure(1:ny-1,:)]...

 .\*[zeros(1,nx);(K(2:ny,:)+K(1:ny-1,:))./2]...

 +[zeros(ny,1),fluid\_pressure(:,2:nx)-fluid\_pressure(:,1:nx-1)]...

 .\*[zeros(ny,1),(K(:,2:nx)+K(:,1:nx-1))./2])...

 + time\_increment.\*fluid\_prod;%factor of 1M because units MPa

 fluid\_mass = max(fluid\_mass + d\_fluid\_mass,0);

 %mass flux from neighbors is the difference in pressure times the perm (times

 %the interfacial area between cells)(times time)

 fluid\_pressure = min(overburden+dstrength\*tensile\_strength\_factor,(defpore./porosity.\*fluid\_mass./defmass-1)./water\_comp);

 if sea\_floor == 1

 fluid\_pressure(1,:) = sea\_floor\_pressure;

 fluid\_mass(1,:) = porosity(1,:).\*block\_area.\*block\_height.\*1100;

 end

 crack\_cells = fluid\_pressure>=overburden+dstrength\*tensile\_strength\_factor;

 fluid\_loss = zeros(ny,nx);

 fluid\_loss(crack\_cells) = fluid\_mass(crack\_cells) - (1+fluid\_pressure(crack\_cells).\*water\_comp).\*defmass.\*porosity(crack\_cells)./defpore;

 if n > burn\_in

 if mod(n,frame\_int) == 0

 loc\_frame(:,:,n/frame\_int+1-burn\_in/frame\_int) = loc(:,:); %Add to the cumulative slip on each cell that failed.

 fluid\_mass\_frame(:,:,n/frame\_int+1-burn\_in/frame\_int) = fluid\_mass(:,:);

 fluid\_pressure\_frame(:,:,n/frame\_int+1-burn\_in/frame\_int) = fluid\_pressure(:,:);

 dstrength\_frame(:,:,n/frame\_int+1-burn\_in/frame\_int) = dstrength(:,:);

 fluid\_loss\_frame(:,:,n/frame\_int+1-burn\_in/frame\_int) = fluid\_loss(:,:);

 end

 end

 fluid\_mass(crack\_cells) = fluid\_mass(crack\_cells) - fluid\_loss(crack\_cells);%leak excess fluid mass out the roof to limit pressure

 loc = loc+time\_increment\*plate\_rate\_persec; %Move the driving plate to the point of the next rupture. Should always be +ive outside rupture loop.

 if flowlaw == 1

 loc = loc-stress.^Fx.\*F.\*exp(-Fe./(R.\*(T+273.15))).\*time\_increment;

 end

 stress = loc+alpha\*([loc(:,1:nx-1)-loc(:,2:nx),zeros(ny,1)]...

 +[loc(1:ny-1,:)-loc(2:ny,:);zeros(1,nx)]...

 +[zeros(1,nx);loc(2:ny,:)-loc(1:ny-1,:)]...

 +[zeros(ny,1),loc(:,2:nx)-loc(:,1:nx-1)]);%Stress on each slider block

 fail = dstrength+non\_asp\_str+0.1\*(overburden-fluid\_pressure+stress/sin(2\*dip)\*(cos(2\*dip)-1))-stress;

 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Wrap up

t = toc;

disp(['Finished calculations in ',num2str(t),' seconds'])

%If there is a burn-in period to remove, take out those results from the

%result vectors.

if burn\_in>0

 rupture\_time = rupture\_time(burn\_in+1:end);

 rupture\_slip = rupture\_slip(burn\_in+1:end);

 rupture\_slip = rupture\_slip(rupture\_time>0);

 slip\_surplus = slip\_surplus(burn\_in+1:end);

 slip\_surplus = slip\_surplus(rupture\_time>0);

 rupture\_area = rupture\_area(burn\_in+1:end);

 rupture\_area = rupture\_area(rupture\_time>0);

 rupture\_init\_X = rupture\_init\_X(burn\_in-1:find(rupture\_init\_X,1,'last')); %did burn\_in -1 to synch up with movie

 rupture\_init\_Y = rupture\_init\_Y(burn\_in-1:find(rupture\_init\_Y,1,'last'));

 rupture\_init\_X = rupture\_init\_X(rupture\_init\_X>0);

 rupture\_init\_Y = rupture\_init\_Y(rupture\_init\_Y>0);

 rupture\_time = rupture\_time(rupture\_time>0);

 rupture\_time = rupture\_time-rupture\_time(1)+1;

 end

 Mw = 2/3\*log10(-rupture\_slip\*block\_area) + 2/3\*log10(32 \* 1e16) - 10.7;

 Mw\_finite = Mw(isfinite(Mw));

 stress\_drop = 32000 \* rupture\_slip./rupture\_area./sqrt(rupture\_area\*block\_area);%megapascales

 slip\_surplus = slip\_surplus./(nx\*ny);

 Mw\_stats(O,P,1:size(Mw\_finite,2)) = Mw\_finite;

 rupture\_time\_stats(O,P,1:size(rupture\_time,2)) = rupture\_time;

 rupture\_slip\_stats(O,P,1:size(rupture\_slip,2)) = rupture\_slip;

 slip\_surplus\_stats(O,P,1:size(slip\_surplus,2)) = slip\_surplus;

 fluid\_prod\_stats(O,P) = sum(sum(fluid\_prod));

 fluid\_loss\_stats(O,P,:) = permute((sum(sum(fluid\_loss\_frame./time\_increment))),[3 2 1]);

 totmass\_stats(O,P,:) = permute(sum(sum(fluid\_mass\_frame(2:end,:,:))),[3 2 1]);

 pgrad\_stats(O,P,:) = permute(sum(fluid\_pressure\_frame(fluidmean,:,:)-fluid\_pressure\_frame(2,:,:))\*1000000/nx/(L\*1000\*fluidmean/ny),[3 2 1]);

 porosity\_frame = defpore - defpore\*minporefactor\*(dstrength\_frame./max\_dstrength);

 porosity\_hist\_stats(O,P,:) = permute(mean(mean(porosity\_frame)),[3 2 1]);

save('results.mat')

 end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Make plots.

if O\*P == 1

%Plot the asperity area and rupture sizes over time.

figure(1)

subplot(2,1,1)

errorbar(rupture\_time(rupture\_time>0),Mw\_finite,Mw\_finite,zeros(size(Mw\_finite)),'.k','CapSize',0)

ylim([min(Mw\_finite)-0.5 max(Mw\_finite)+0.5])

ylabel('Magnitude')

subplot(2,1,2)

plot(rupture\_time,slip\_surplus)

xlabel('Time (yr)')

ylabel('Average slip deficit (m)')

figure(2)

X = min(Mw\_finite):0.1:max(Mw\_finite);

for i = 1:size(X,2)

 Y(i) = sum(Mw\_finite>X(1,i));

end

plot(X,log10(Y),'Marker','o','MarkerEdgeColor',[1 0 0],'LineStyle','none')

xlabel('Magnitude')

ylabel('log\_{10} Cumulative number')

axis equal

end

save(num2str(cc(1:5)),'defperm','defpore','dip','Ea','A','G','C','fluid\_prod\_mag','minKfactor','minporefactor','alpha','phi','sea\_floor')