Table of Contents


```
function impactorTempMeltFunc(fn,eta_0,E_a)

    %{
    Function to evolve impact melt chambers on Europa. Simulations end
    when there is a negligible amount of melt left from the impact.
 The
    ouputs are saved at the breakpoint in line 294. This model is
 meant for
    short term simulations, on the order of the sinking of impact
 melts.

    Variables:
        fn (string): is the directory to find the initial conditions
 (the
        outputs from the impact simulation)
        eta_0 (int): basal viscosity of ice in Pa s, 1e14 is a common
 value
        E_a (int): visocisty activation energy in Arrhenihus
 relationship
        given in J/mol, 50e3 is a common value

    Author: Evan Carnahan, evan.carnahan@utexas.edu, 11/20/2022
    %}

    set(groot,'defaulttextinterpreter','latex')
    set(groot,'defaultAxesTickLabelInterpreter','latex')
    set(groot,'defaultLegendInterpreter','latex')
    warning off; % matrix is close to singular due to viscosity
 contrast
```

Load initial condition to be evolved

make ice shell thickness based on impact code passed

```
    if any([all(fn == '03321') all(fn == '03800') all(fn == '04304')])
        d = 10*1e3; % m
    elseif any([all(fn == '03314') all(fn == '03402') all(fn
 == '03400')])
        d = 20*1e3; % m
    elseif any([all(fn == '03313') all(fn == '03701')])
        d = 30*1e3; % m
    elseif all(fn == '03330')
        d = 40*1e3; % m
    end

    % threshold of intial fluid left in ice shell to stop simulation
 at
    termFrac = 0.005;

    % load simulations from initial conditions folder
    fp = '../initial_conditions/ic';
    load([fp fn '_100.mat'],'T','phi');
```

```
Not enough input arguments.

Error in impactorTempMeltFunc (line 24)
    if any([all(fn == '03321') all(fn == '03800') all(fn == '04304')])
```

Set physical parameters and make dimension-less scales

physical parameters for ice

```
    T_t = 100; % surface temperature
    a = 185; % ice specific heat parameter
    b = 7.037; % ice specific heat parameter
    T_b = 273; % melting temperature, K
    rho_i = 920; % ice density at melting temperature, kg/m^3
    grav = 1.315; % Europa gravity, m/s^2
    DT = T_b - T_t; % difference in temperature
    alpha = 1.6e-4; % thermal expansion coefficient, 1/K
    R = 8.314; % universal gas constant, J K^-1 mol^-1
    Apar = E_a/R/T_b; % viscosity exponenet

    % physical properties of water
    latHeat = 334e3; % latent heat of fusion, J/kg
    rho_w = 1e3; % density of water, kg/m^3
    c_pw = 4.186e3; % specific ehat of water, J/kg
    kappa_w = 0.56; %thermal diffusivity of water, W/(m K)
    porViscPar = 45;

    % temperature and melt fraction dependent viscosity, Pa s
    barrViscPhi = @(nonT,phi) max(exp(Apar*(T_b./(DT.*nonT
+T_t)-1)).*exp(-porViscPar*phi),1e-2);
    c_fun = @(nonT) a+b*(DT*nonT+T_t); %J/(kg K)
    kappa_b = 3.3; %thermal conductivity of ice, set to be consistent
 with Cox and Bauer, 2015
```

```
    D_T = kappa_b/(rho_i *c_fun(1)); % thermal diffusicvity of ice,
 m^2/s
    c_pi = c_fun(1); %constant specific heat, J/(kg K)


    % thershold for melting: dimensionless boundary between partial
 and
    % fully melted regions
    mixZone = (rho_w*latHeat)/(rho_i*c_pi*DT);

    phi_fun = @(nonH) nonH * (rho_i*c_pi*DT)/(rho_w*latHeat);
    TWater_fun = @(nonH) nonH * (rho_i*c_pi)/(rho_w*c_pw) - latHeat/
(DT*c_pw) + 1;
    compBouy_fun = @(phi)  phi*rho_i*(rho_w/rho_i-1)*(grav*d^3/
(eta_0*D_T));

    % conversion in dimensionless units with constant specific heat
    nonH_fun = @(nonT) nonT - 1;
    nonT_fun = @(nonH) nonH + 1;
    % condictivity is weighted average of mixture components
    porKappaPrime_fun = @(phi,nonT) (phi*kappa_w + (1-phi).*kappa_b)/
kappa_b;
    porNonH_fun = @(phi,nonT) (1-phi).*(nonH_fun(nonT)) + ...
        (phi*rho_w)./(rho_i*c_pi*DT).*(latHeat+c_pw*DT*(nonT-1));

    % characteristic scales for general convection
    t_c = d^2/D_T; % dimensionless time, s
    Ra = rho_i*grav*alpha*d^3*DT/(eta_0*D_T); % basal Rayleigh number

    % non-dimensionalize tmeprature
    T = (T - T_t)/DT;

    % thermal condutvity in convective ocean, set to maintain vertical
    % geotherm in ocean
    kappa_c = 100;
```

build cylindirical grid for numerical solution

build grid

```
    grRes = 100; % grid resolution
    ocTh = grRes/5; % make ocean below the ice shell
    Gridp.xmin = 0; Gridp.xmax = 2; Gridp.Nx = grRes;
    Gridp.ymin = -ocTh/grRes; Gridp.ymax = 1; Gridp.Ny = grRes+ocTh;
    Gridp.geom = 'cylindrical_rz';
    Grid = build_stokes_grid_cyl(Gridp);

    % convert inital condition to grid
    TGr = reshape(T,grRes,Grid.p.Nx);
    phiGr = reshape(phi,grRes,Grid.p.Nx);
    % get initial melt volumes
    phiOrig = sum(sum(phiGr(10:end,:),1).*Grid.p.V(Grid.p.dof_ymin)' *
 d^3);
```

```
% build ocean layer
TOc = ones(ocTh,Grid.p.Nx);
phiOc = ones(ocTh,Grid.p.Nx);

% combine ice shell and ocean
TGr = [TOc; TGr];
phiGr = [phiOc; phiGr];
T = TGr(:);
phi = phiGr(:);
H = porNonH_fun(phi,T);
```

build operators

```
Zp = zeros(Grid.p.N);
Ip = speye(Grid.p.N);
[D,Edot,Dp,Gp,I,Gyy]=build_stokes_ops_cyl(Grid);
linInds = find(Gyy > 0);
[row,~] = ind2sub(size(Gyy),linInds);
[X,Y] = meshgrid(Grid.p.xc,Grid.p.yc);
```

Build boundary conditions for temperature and flow equation

Fix temperature at top of ice shell with Dirchlet BC

```
T0 = 0;
H0 = nonH_fun(T0);
Param = struct('H',{},'g',{},'dof_dir',{});
Param(1).H = struct('dof_dir',{},'dof_f_dir',{},'g',{},'dof_neu',
{},'dof_f_neu',{},'qb',{},'dof_out',{});

% fix bottom heat flux, Neumann BC, to maintain linear geotherm in
 ice shell
qPrime = 1;
Param.H(1).dof_dir = [Grid.p.dof_ymax];
Param.H(1).dof_f_dir = [Grid.p.dof_f_ymax];
Param.H(1).g = [H0*ones(length(Grid.p.dof_ymax),1)];

Param.H(1).dof_neu =
 [Grid.p.dof_xmin;Grid.p.dof_xmax;Grid.p.dof_ymin];
Param.H(1).dof_f_neu =
 [Grid.p.dof_f_xmin;Grid.p.dof_f_xmax;Grid.p.dof_f_ymin];
Param.H(1).qb =
 [0*Grid.p.dof_f_xmin;0*Grid.p.dof_f_xmax;qPrime*ones(size(Grid.p.dof_f_ymin))];
[BH,NH,fn_H] = build_bnd(Param.H,Grid.p,Ip);
Param.H(1).dof_out = [Grid.p.dof_ymin];


% Free slip boundary condition for Stokes equation
Param(1).dof_dir =  [...
                  Grid.x.dof_xmax;...           %set x_max x-vel
```

```
                  Grid.x.dof_xmin;...          %set x_min x-vel
                  Grid.x.N+Grid.y.dof_ymin;... %set y_min y-vel
                  Grid.x.N+Grid.y.dof_ymax;... %set y_max y-vel
                  Grid.p.Nf+1];                %set pressure
Param(1).g = 0*Param.dof_dir;
Param.g(end) = 0;
B = I([Param.dof_dir],:);
N = I;
N(:,[Param.dof_dir]) = [];
fs_T = nan(size(T));


% create aarrays for time evolution storage
it = 1e9;
netMelt = [];
phiDrain1 = 0;
phiDrain2 = 0;
tTot = 0;
tVec = [];
phiDrain1Vec = [];
phiDrain2Vec = [];
phiFracRem = [];
```

temporal evolution

```
for i = 1:it

    % calculate porosity from enthalpy
    [T,phi] = enthMelt(H,mixZone,nonT_fun,phi_fun,TWater_fun);
    H = porNonH_fun(phi,T);
```

Bouyancy force as the body force to Stokes flow

```
    %temp bouyancy
    Tplot= reshape(T,Grid.p.Ny,Grid.p.Nx);
    Tdiag = comp_mean(Tplot,1,1,Grid.p);
    Tvec = diag(Tdiag);
    Ty = Tvec(Grid.p.Nfx+1:Grid.p.Nf);
    fs_T(Ty>1) = -Ra*1;
    fs_T(Ty<=1) = -Ra*Ty(Ty<=1);

    %compositional bouyancy
    phiPlot= reshape(phi,Grid.p.Ny,Grid.p.Nx);
    phiDiag = diag(comp_mean(phiPlot,1,1,Grid.p));
    phiY = phiDiag(Grid.p.Nfx+1:Grid.p.Nf);
    fs_por = compBouy_fun(phiY);

    % higher porosity acts against Ra bouyancy force
    fsVec = fs_T + fs_por;
    fs = [zeros(Grid.p.Nfx,1); fsVec; zeros(Grid.p.N,1)];
```

```
%Gxx variable viscosity matrix
nxxVec = zeros(Grid.x.Nfx,1);
nxxVec(Grid.x.Ny+Grid.p.dof) = Tplot;

%Gyy variable viscosity matrix
nyyVec = zeros(Grid.y.Nfy,1);
nyyVec(row) = Tplot;
ncVec = comp_mean_corners(Tplot,-1,Grid.p);
tempVec = [nxxVec;nyyVec;ncVec];
```

Porosity and temperature dependent viscosity

porosity dependent viscosity

```
phiPlot = reshape(phi,Grid.p.Ny,Grid.p.Nx);
nxxVecPhi = zeros(Grid.x.Nfx,1);
nxxVecPhi(Grid.x.Ny+Grid.p.dof) = phiPlot;

%Gyy variable viscosity matrix
nyyVecPhi = zeros(Grid.y.Nfy,1);
nyyVecPhi(row) = phiPlot;

ncVecPhi = comp_mean_corners(phiPlot,-1,Grid.p);
phiVec = [nxxVecPhi;nyyVecPhi;ncVecPhi];

% merge temperature and melt fraction viscosities
viscVec = barrViscPhase(tempVec,phiVec,barrViscPhi);
viscVec(isnan(viscVec)) = 0;
viscMat = spdiags(viscVec,0,length(viscVec),length(viscVec));
```

Stokes Flow calcualtion

make linear operators

```
        tau = D*2*viscMat*Edot;
        L = [tau,Gp;
            Dp,Zp];
        % solve for flow velocities
        u = solve_lbvp(L,fs,B,Param.g,N);
        vx = u(1:Grid.p.Nfx);
        vy = u(Grid.p.Nfx+1:(Grid.p.Nfx+Grid.p.Nfy));
        vm = [vx;vy];
        vmax= max(abs(vm));
        % Adaptive time stepping based on CFL condition
        dt = min([0.5*Grid.p.dx^2/kappa_c, Grid.p.dx/
vmax,0.5*Grid.p.dy^2/kappa_c, Grid.p.dy/vmax])*0.8;
        p  = u((Grid.p.Nfx+Grid.p.Nfy+1):end);
```

non-linear thermal conducitivity matricies

```
kappaPrime = porKappaPrime_fun(phi,T);
% select near bouundary ocean cells
```

```
ocLog = Y(:) < 2/grRes & phi > 0.5;
kappaPrime(ocLog) = kappa_c;
kappaPrimePlot = reshape(kappaPrime,Grid.p.Ny,Grid.p.Nx);
kappaFace = comp_mean(kappaPrimePlot,1,1,Grid.p);
```

Advection of enthalpy, diffusion of temperature

```
AH = build_adv_op(vm,H,dt,Gp,Grid.p,Param.H,'mc');
L_T_I = Ip;
L_T_E_T = - dt*(-Dp*kappaFace*Gp);
L_T_E_H = Ip - dt*(Dp*AH);
RHS_T = L_T_E_H*H + L_T_E_T*T + (fn_H)*dt;
H = solve_lbvp(L_T_I,RHS_T,BH,Param.H.g,NH);
```

calculate net melt and melt transported to "ocean"

make two planes to measure the melt transported through plane 1

```
        abInd = 5;
        phiGr = reshape(phi,Grid.p.Ny,Grid.p.Nx);
        bdPhi = mean(phiGr(ocTh+abInd:ocTh+abInd
+1,:),1).*Grid.p.V(Grid.p.dof_ymin)';
        vyGr = reshape(vy,Grid.y.Ny,Grid.y.Nx);
        bdVy = vyGr(ocTh+abInd+1,:);
        meltTrans = bdPhi .* bdVy / Grid.p.dy * dt * d^3;
        phiDrain2 = phiDrain2 - sum(meltTrans(bdVy < 0)); % m^3
        phiDrain2Vec = [phiDrain2Vec phiDrain2];

        % plane 2
        phiGr = reshape(phi,Grid.p.Ny,Grid.p.Nx);
        bdPhi = mean(phiGr(ocTh+(abInd-1):ocTh
+(abInd-1)+1,:),1).*Grid.p.V(Grid.p.dof_ymin)';
        vyGr = reshape(vy,Grid.y.Ny,Grid.y.Nx);
        bdVy = vyGr(ocTh+(abInd-1)+1,:);
        meltTrans = bdPhi .* bdVy / Grid.p.dy * dt * d^3;
        phiDrain1 = phiDrain1 - sum(meltTrans(bdVy < 0)); % m^3
        phiDrain1Vec = [phiDrain1Vec phiDrain1];

        % total melt left
        netMelt = [netMelt sum(phi .* Grid.p.V * d^3)];
        % select near bouundary ocean cells
        % time stepping
        tTot = tTot + dt*t_c/(3.154e7); %Yrs
        tVec = [tVec tTot];

        % calculate remaining phi above near ocean, bottom 20% of ice
 shell
        phiRem = sum(sum(phiGr(ocTh
+grRes/5:end,:),1).*Grid.p.V(Grid.p.dof_ymin)' * d^3);
        phiFracRem = [phiFracRem phiRem/phiOrig];
```

```
        % condition for ending simulation
        if phiFracRem(end) < termFrac || (i > 1000 && phiFracRem(end)
> phiFracRem(end-1))
            % save point
            save(['impact_' fn '_eta0_' num2str(log10(eta_0)) '_Ea_'
num2str(E_a/1e3) '_output.mat'],...

 'Tplot','phi','Grid','phiDrain1Vec','phiDrain2Vec','phiOrig','tVec',...

 'phiFracRem','T','phi','tVec','phiDrain1Vec','phiDrain2Vec','phiOrig')
            break
        end
```

PLOTTING

```
         if mod(i,10) == 0
             i

            figure(4);
            [PSI,psi_min,psi_max] = comp_streamfun(vm,Grid.p);
            set(gcf, 'Position', [50 50 1500 600])
            subplot(3,3,1)
            cla;
            hold on
            axis equal
            contourf(X*d/1e3,Y*d/1e3-Grid.p.dy,Tplot*DT
+T_t,40,'linestyle','none'),view(2),hold on
            c = colorbar('NorthOutside');
            caxis([min(Tplot(:)) max(Tplot(:))]*DT+T_t);
            cStrVal = linspace(min(PSI(:)),max(PSI(:)),10);

 contour(Grid.p.xf*d/1e3,Grid.p.yf*d/1e3,PSI,'k','LevelList',cStrVal);
            c.Label.String = 'Temperature, K';
            xlabel('x-dir, km')
            ylabel('z-dir, km')

            subplot(3,3,2)
            cla;
            axis equal
            hold on

 contourf(X,Y,reshape(phi,Grid.p.Ny,Grid.p.Nx),40,'linestyle','none'),view(2),hold
            c = colorbar('NorthOutside');

 contour(X,Y,reshape(phi,Grid.p.Ny,Grid.p.Nx),'r','LevelList',5e-2)
            xlabel('x-dir, 1')
            ylabel('z-dir, 1')
            c.Label.String = 'Melt fraction, 1';


            subplot(3,3,3)
            cla;
            plot(mean(reshape(phi,Grid.p.Ny,Grid.p.Nx),2),Grid.p.yc)
```

```
            ylabel('z-dir, 1');
            xlabel('Average melt fraction');

            subplot(3,3,4)
            cla;
            plot(mean(reshape(T,Grid.p.Ny,Grid.p.Nx),2),Grid.p.yc)
            xlabel('Avg. temp');
            ylabel('z-dir, 1');


            subplot(3,3,5)
            cla;
            hold on
            plot(phiFracRem);
            ylabel('Total melt remaining, \%');
            xlabel('Time, yrs')

            subplot(3,3,6)
            cla;
            hold on
            axis equal

contourf(X,Y,reshape(kappaPrimePlot,Grid.p.Ny,Grid.p.Nx),40,'linestyle','none'),v
            c = colorbar('NorthOutside');
            xlabel('x-dir, m')
            ylabel('z-dir, m')
            c.Label.Interpreter = 'latex';
            c.TickLabelInterpreter = 'latex';
            c.Label.String = 'Non-dim thermal conductivity';

            [Xc,Yf] = meshgrid(Grid.p.xc,Grid.p.yf);
            subplot(3,3,7)
            cla;
            plot(tVec,phiDrain1Vec);
            hold on
            plot(tVec,phiDrain2Vec);
            legend('intefrace','1 above
interface','Location','NorthWest');
            ylabel('Melt drained, m$^3$');
            xlabel('Time, yrs')

            subplot(3,3,8)
            cla;
            plot(tVec,phiDrain1Vec/phiOrig*100);
            hold on
            plot(tVec,phiDrain2Vec/phiOrig*100);
            legend('interface','1 above
interface','Location','NorthWest');
            ylabel('Percentage of melt drained, \%');
            xlabel('Time, yrs')

            subplot(3,3,9)
            cla;
            axis equal
```

```
            contourf(Xc,Yf,reshape(-fsVec,Grid.p.Ny
+1,Grid.p.Nx),40,'linestyle','none'),view(2),hold on
            c = colorbar('NorthOutside');
            c.Label.Interpreter = 'latex';
            c.TickLabelInterpreter = 'latex';
            c.Label.String = 'Total Bouyancy, 1';
            xlabel('x-dir, 1')
            ylabel('z-dir, 1')

         end

     end

end
```

Published with MATLAB® R2021a