```
function
[BetaFinal,TFinal,ThetaFinal,GammaFinal,TimeFinal]=duffsim(MoistureContent,MineralContent,BulkDensity
,PlotsOn)
%Duff smoldering simulation model by Chris Dugaw
%function
%[BetaFinal,TFinal,ThetaFinal,GammaFinal,TimeFinal]/=duffsim(MoistureContent,MineralContent,
BulkDensity,PlotsOn)
%Inputs
%   MoistureContent - Gravametric Moisture Content (Gamma)
%   MineralContent - Gravametric Mineral Content (Alpha)
%   BulkDensity - kg/m^3 (Rho)
%   PlotsOn - 1 gives plots 0 surpresses plot output
%Outputs
%   BetaFinal - Matrix of Burn State 0->Unburned 1->Burning 2->Burned
%   TFinal - Matrix of Final Temperature (Kelvin)
%   ThetaFinal - Final Volumetric Moisture Content(m^3/m^3)
%   GammaFinal - Final Gravametric Moisture Content (kg/kg)
%   TimeFinal - Elapsed Time (hours)


warning('error','MATLAB:singularmatrix');

if MineralContent >= 1 || MineralContent < 0
    error('Mineral Content must be strictly less than 1 and greater than 0')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Intialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Set Random Number Generator Seed %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    TimeSeed=round(1e+9*mod(1e+7*now,1));
    s = RandStream.create('mt19937ar','seed',TimeSeed);
          RandStream.setDefaultStream(s);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Set Spatial and Temporal Dimensions%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    DeltaX=0.075; %meters

    LengthX=1.5;  %meters
    LengthY=0.6;  %meters

    NumCols=ceil(LengthX/DeltaX);
    NumRows=ceil(LengthY/DeltaX);

    NumCells=NumRows*NumCols;

    DeltaT=0.005;  %Time Step Hours
    MaxTime = 30; %hours
    MaxTSteps=ceil(MaxTime/DeltaT);

    Active=true; %Booleon Variable true->Active Smoldering
    Time=0;
    Step=0;

    %Initalize Functions for Smolder Velocity Fit

    Lambda=0.05; %Uninhibited smolder Velocity (m/hr)

```



```
    MeanBurnTimeSteps=round(DeltaX/Lambda/DeltaT);
    BurnTimeSteps=MeanBurnTimeSteps*ones(NumRows,NumCols);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Initialize Physical Parameters%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    WaterMolecularWeight=0.018; %Water MW (kg/mol)
    GasConstant=8.314;          %Universal Gas Constant J/(mol K)

    Dv0=0.07632; %Diffusivity of water in air at STP 2.12E-5 (m^2/s) x 3600
                 %Campbell 1994

    Xi=0.66; %Tortuosity Correction (unitless) Campbell 1995 pg. 365
    Eta=1; %Vapor flow enhancement factor (unitless)
    Q0=1.97; %Used in thermal conductivity calculations (unitless)
             % Value for Peat Moss Taken from Table 2 Campbell et al. 1994

    %Densities from Ghildayl & Tripathi 1987 Table 18.1 pg. 512
    RhoWater=1000; %(kg/m^3)
    RhoMineral=2650; %(kg/m^3) Silica / Soil Minerals
    RhoOrganic=1300; %(kg/m^3)

    %Specific Heats from Ghildayl & Tripathi 1987 Table 18.1 pg. 512
    cWater=4186; %J/(kg K)
    cMineral=795;  %J/(kg K)
    cOrganic=1924; %J/(kg K)

    %Convective Loss Rates
    ConvectiveCooling=1e+5; %Should be on order of magnitude of heat capcity
    ConvectiveVaporLoss=100;

    %Pressures
    AmbientPressure=101325; %(Pa)
    StandardPressure=101325; %(Pa)

    %Temperatures (degrees Kelvin)
    StandardTemp=273.15;
    AmbientTemp=293.15;
    BurnTemp=673.15;

    %Ambient Humidity and Partial Pressure of Water Campbell et al. 1995
    SAmbient=1-373.15*AmbientTemp.^(-1);
    AmbientPSat=101325*exp(13.3016*SAmbient-2.042*SAmbient.^2+0.26*...
        SAmbient.^3+2.69*SAmbient.^4);
    AmbientHumidity=0.80;
    AmbientVaporPp=AmbientHumidity*AmbientPSat;

    %ThermalConductivities from Ghildayl & Tripathi 1987 Table 18.1 pg. 512
    lambda_o=903600; %251x3600 (J/hr)
    lambda_m=10544400; %2929*3600 (J/hr)

    %Shape Factors
    ShapeFactorOrganic=0.33; %Campbell 1994 Peat Moss
    ShapeFactorMineral=0.1;  %Approx mean of mineral soils from Campbell 1994

    Alpha=MineralContent*ones(NumRows,NumCols);
    Rho=BulkDensity*ones(NumRows,NumCols);
    Delta=Rho.*(1-Alpha).^(-1);
    PhiOrganic=Rho/RhoOrganic;
    PhiMineral=Alpha.*Delta/RhoMineral;
    Porosity=1-PhiMineral-PhiOrganic;
    ThetaMin=0;
```

```
127     Theta0=0.17; %Value for Peat from Table 2 Campbell 1994 x_{wo}
128
129     %Variables for Soil-Water Potential Curve
130     Matric0=-1e6; %Matric potential at Theta=0;
131     LogNegMatric0=log(-Matric0); %Campbell 1995 p.364
132     ThetaAirDry=0.0317;
133     %m^3/m^3 extroplated from mean of values in Campbel 1995 Table 1
134     Theta1=6.3*ThetaAirDry;
135     MatricMin=VMC2Matric(ThetaMin);
136     BurnMatric=3*MatricMin;  %Needs to be small enough so
137                              %that Vapor Pressure in Burning Cells is less
138                              %than Ambient Pressure
139
140
141     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
142     %Frandsen Inigition Probability Coefficents%
143     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
144
145     FrandsenProb=[];
146
147     %From Frandsen 1997 Spruce Pine Duff
148     % Moisture and content content given as fraction and not percent.
149     B0=58.6921 - 3; %Reduced slightly to account for predrying
150     B1=-27.37; %-0.2737*100 convert fraction to percent.
151     B2=-54.13; %-0.5413*100 convert fraction to percent.
152     B3=-0.1246;
153
154     %%%%%%%%%%%%%%%%%%%%%%%%%%%%
155     %Initialize State Variables%
156     %%%%%%%%%%%%%%%%%%%%%%%%%%%%
157
158     BetaN=zeros(NumRows,NumCols); %Burn State
159     BetaNp1=zeros(NumRows,NumCols); %step next step n+1
160     TN=ones(NumRows,NumCols); %Temp
161     TNp1=ones(NumRows,NumCols); %step next step n+1
162     Gamma=MoistureContent*ones(NumRows,NumCols); %Gravametric Moist. Cont.
163     ThetaN=GMC2VMC(Gamma);%Volumetric moisture content
164     ThetaNp1=ones(NumRows,NumCols); %step next step n+1
165     MatricN=VMC2Matric(ThetaN);%Matric Potential
166     MatricNp1=ones(NumRows,NumCols); %step next step n+1
167     PhiAir=zeros(NumRows,NumCols); %Volumetric Air content
168     HeatCapacity=zeros(NumRows,NumCols); %Specific Heat
169     HeatVaporization=zeros(NumRows,NumCols); %Latent Heat of Vaporization
170     PSat=zeros(NumRows,NumCols); %Saturation Vapor Pressure
171     WaterVaporPpNp1=zeros(NumRows,NumCols); %Water Vapor Partial Pressure
172     OverPressure=[]; %Indicies of cells were Water Vapor Partial Pressure
173                      %is predicted to exceed ambient pressure
174     humidity=zeros(NumRows,NumCols);
175     VaporConductivity=zeros(NumRows,NumCols);
176     ThermalConductivity=zeros(NumRows,NumCols);
177     Steffan=zeros(NumRows,NumCols);
178
179
180     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181     %Variables for Newton's Method%
182     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
183
184     EpsHeat=3.6e+4 * NumCells; %From Campbell 1995 (p. 366) 100 W/m^2 = 100 J/(s m^2)
185                      % equivalent to 3.6e+4 J/(hr m) assuming 10cm depth
186     EpsWater=0.0036 * NumCells; %From Campbell 1995 (p. 366) 1e-5 kg/(m^2 s)
187                       % equivelent to 0.0036 kg/(m hr)
188
189     ErrVector=sparse(2*NumCells,1);
```



```
190     PartialsMatrix=sparse(2*NumCells,2*NumCells);
191     SolvingCells=[];
192     SolvingCellsDoubled=[];
193     NumSolvingCells=0;
194
195     dPSat_dT=zeros(NumRows,NumCols);
196     dHeatVaporization_dT=48/WaterMolecularWeight;
197
198     VaporLossCoef=DeltaX^2*WaterMolecularWeight/GasConstant*...
199             ConvectiveVaporLoss;
200
201     maxits=500; %Maximum number of interations for Newton's method
202     its=0;
203
204     Initialize();
205
206
207 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
208 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
209
210 %%%%%%%%%%
211 %Main Loop%
212 %%%%%%%%%%
213
214 while Active && Step <= MaxTSteps
215     Time=Time+DeltaT;
216     Step=Step+1;
217
218     UpdateBeta()  %Determines BetaNp1
219     BurningCells=(BetaNp1==1 | BetaNp1==0.9);
220     NumBurningCells=sum(sum(BurningCells));
221
222     if NumBurningCells>0 %Active?
223
224         SetSolvingCells()
225
226         %Take Explicit Step
227         TNp1=TN;
228         MatricNp1=MatricN;
229         TNp1(BurningCells)=BurnTemp;
230         MatricNp1(BurningCells)=BurnMatric;
231         Set_BC();
232         ThetaNp1=Matric2VMC(MatricNp1);
233
234         UpdateVapor()
235         UpdateConductivities()
236
237         % Uncomment this to take explicit step
238          Explicit_Step()
239          Set_BC_Explicit()
240
241         %Prep. for Newton's Method Implicit Scheme
242         MatricNp1=VMC2Matric(ThetaNp1);
243         UpdateVapor()
244         UpdateConductivities()
245
246         converged=false;
247         its=1;
248
249         Compute_Errs()
250
251         while (~converged && its<maxits)
252             Compute_Partials()
```

```
253
254             try
255                 DeltaVector=-PartialsMatrix\ErrVector;
256             catch
257                 warning('Singular Partials Matrix. Reduce time step size (DeltaT)')
258             end
259
260
261             DeltaTemp=zeros(1,NumCells);
262             DeltaTemp(SolvingCells)=DeltaVector(1:NumSolvingCells);
263             DeltaTemp=full(reshape(DeltaTemp,NumRows,NumCols));
264
265             DeltaMatric=zeros(1,NumCells);
266             DeltaMatric(SolvingCells)=DeltaVector(NumSolvingCells+1:end);
267             DeltaMatric=full(reshape(DeltaMatric,NumRows,NumCols));
268
269             MatricNp1=MatricNp1+DeltaMatric;
270             MatricNp1(MatricNp1>0)=-0.001; %Limit Like Campbell
271
272             TNp1=TNp1+DeltaTemp;
273
274
275             Set_BC();
276
277             TotErrHeat=sum(abs(ErrVector(1:NumSolvingCells)));
278             TotErrWater=sum(abs(ErrVector(NumSolvingCells+1:end)));
279
280             if TotErrHeat<EpsHeat && TotErrWater<EpsWater
281                 converged=true;
282                %disp('******************** Converged ********************');
283             else
284                 its=its+1;
285                 if its==maxits-1;
286                     its=1;
287                     %disp('####################### No Convergence  ########################');
288                     TNp1(SolvingCells)=TNp1(SolvingCells)+10*randn(NumSolvingCells,1);
289                     MatricNp1(SolvingCells)=MatricNp1(SolvingCells)+...
290                         1e+5*randn(NumSolvingCells,1);
291                 end
292             end
293
294             ThetaNp1=Matric2VMC(MatricNp1);
295             UpdateVapor()
296             Compute_Errs()
297
298         end
299
300         MatricN=MatricNp1;
301         TN=TNp1;
302         BetaN=BetaNp1;
303         ThetaN=Matric2VMC(MatricN);
304         Gamma=VMC2GMC(ThetaN);
305     else %No Burning cells
306         Active=false;
307     end
308
309      if PlotsOn && mod(Step,floor(0.1/DeltaT))==0
310          make_plots()
311     end
312 end
313
314 %Cleave off boundary cells and output
315 TFinal=TNp1(2:end-1,2:end-1);
```



```
316 BetaFinal=round(BetaNp1(2:end-1,2:end-1));
317 ThetaFinal=ThetaNp1(2:end-1,2:end-1);
318 GammaFinal=Gamma(2:end-1,2:end-1);
319 TimeFinal=Time;
320
321 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
322
323     function []=Initialize()
324         %Edge Ignition (Bottom)
325         %BetaN(2,2:end-1)=1;
326
327         %Edge Ignition (Left)
328         BetaN(2:end-1,2)=0.9;
329
330
331         %Point Ignition
332         %BetaN(round(NumRows/2),round(NumCols/2))=1;
333
334         BetaN(:,[1,end])=3;
335         BetaN([1,end],:)=3;
336         BurningCells=(BetaN==0.9);
337         TN=AmbientTemp*ones(NumRows,NumCols);
338         TN(BurningCells)=BurnTemp;
339         MatricN(BurningCells)=BurnMatric;
340
341     end
342
343 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
344
345     function []=Explicit_Step()
346         NewT=zeros(NumRows,NumCols);
347         NewTheta=zeros(NumRows,NumCols);
348
349
350         NewTheta(2:end-1,2:end-1)= -DeltaT/(RhoWater*DeltaX^2)*(...
351             WaterVaporPpNp1(2:end-1,2:end-1)/2.*...
352             (VaporConductivity(1:end-2,2:end-1)+...
353             VaporConductivity(3:end,2:end-1)+...
354             VaporConductivity(2:end-1,1:end-2)+...
355             VaporConductivity(2:end-1,3:end)+...
356             4*VaporConductivity(2:end-1,2:end-1)...
357             ) - ...
358             WaterVaporPpNp1(1:end-2,2:end-1)/2.*...
359             (VaporConductivity(1:end-2,2:end-1)+...
360             VaporConductivity(2:end-1,2:end-1))-...
361             WaterVaporPpNp1(3:end,2:end-1)/2.*...
362             (VaporConductivity(3:end,2:end-1)+...
363             VaporConductivity(2:end-1,2:end-1))-...
364             WaterVaporPpNp1(2:end-1,1:end-2)/2.*...
365             (VaporConductivity(2:end-1,1:end-2)+...
366             VaporConductivity(2:end-1,2:end-1))-...
367             WaterVaporPpNp1(2:end-1,3:end)/2.*...
368             (VaporConductivity(2:end-1,3:end)+...
369             VaporConductivity(2:end-1,2:end-1))...
370             )+...
371             ThetaNp1(2:end-1,2:end-1)...
372             -DeltaT/RhoWater*ConvectiveVaporLoss*WaterMolecularWeight*...
373             (Porosity(2:end-1,2:end-1)-ThetaN(2:end-1,2:end-1))./...
374             Steffan(2:end-1,2:end-1).*...
375             (WaterVaporPpNp1(2:end-1,2:end-1)./...
376             (GasConstant*TNp1(2:end-1,2:end-1))-...
377             AmbientVaporPp/(GasConstant*AmbientTemp)...
378             );
```

```
379         %Note WaterVaporPpNp1 is same as WaterVaporN at this point in code
380
381
382         ThetaNp1(SolvingCells)=NewTheta(SolvingCells);
383
384         NewT(2:end-1,2:end-1)=-DeltaT/(DeltaX^2)*(TNp1(2:end-1,2:end-1)/2.*...
385             (ThermalConductivity(1:end-2,2:end-1)+...
386             ThermalConductivity(3:end,2:end-1)+...
387             ThermalConductivity(2:end-1,1:end-2)+...
388             ThermalConductivity(2:end-1,3:end)+...
389             4*ThermalConductivity(2:end-1,2:end-1)...
390             ) - ...
391             TNp1(1:end-2,2:end-1)/2.*...
392             (ThermalConductivity(1:end-2,2:end-1)+...
393             ThermalConductivity(2:end-1,2:end-1))-...
394             TNp1(3:end,2:end-1)/2.*...
395             (ThermalConductivity(3:end,2:end-1)+...
396             ThermalConductivity(2:end-1,2:end-1))-...
397             TNp1(2:end-1,1:end-2)/2.*...
398             (ThermalConductivity(2:end-1,1:end-2)+...
399             ThermalConductivity(2:end-1,2:end-1))-...
400             TNp1(2:end-1,3:end)/2.*...
401             (ThermalConductivity(2:end-1,3:end)+...
402             ThermalConductivity(2:end-1,2:end-1)))...
403             ./HeatCapacity(2:end-1,2:end-1)+...
404             TNp1(2:end-1,2:end-1)...
405             +RhoWater*HeatVaporization(2:end-1,2:end-1)...
406             ./HeatCapacity(2:end-1,2:end-1).*...
407             (ThetaNp1(2:end-1,2:end-1)-ThetaN(2:end-1,2:end-1))...
408             -DeltaT*ConvectiveCooling*(TNp1(2:end-1,2:end-1)-AmbientTemp)...
409             ./HeatCapacity(2:end-1,2:end-1);
410
411         TNp1(SolvingCells)=NewT(SolvingCells);
412
413     end
414
415 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
416
417     function []=Set_BC_Explicit()
418         %No Flux BC
419         TNp1(1,:)=TNp1(2,:);
420         TNp1(end,:)=TNp1(end-1,:);
421         TNp1(:,1)=TNp1(:,2);
422         TNp1(:,end)=TNp1(:,end-1);
423
424         ThetaNp1(1,:)=ThetaNp1(2,:);
425         ThetaNp1(end,:)=ThetaNp1(end-1,:);
426         ThetaNp1(:,1)=ThetaNp1(:,2);
427         ThetaNp1(:,end)=ThetaNp1(:,end-1);
428
429     end
430
431 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
432
433     function []=Set_BC()
434         %No Flux BC
435         TNp1(1,:)=TNp1(2,:);
436         TNp1(end,:)=TNp1(end-1,:);
437         TNp1(:,1)=TNp1(:,2);
438         TNp1(:,end)=TNp1(:,end-1);
439
440         MatricNp1(1,:)=MatricNp1(2,:);
441         MatricNp1(end,:)=MatricNp1(end-1,:);
```



```
442         MatricNp1(:,1)=MatricNp1(:,2);
443         MatricNp1(:,end)=MatricNp1(:,end-1);
444
445     end
446
447
448 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
449
450     function []=Compute_Errs()
451         ErrHeat=zeros(NumRows,NumCols);
452         ErrWater=zeros(NumRows,NumCols);
453
454         ErrHeat(2:end-1,2:end-1)=TNp1(2:end-1,2:end-1)/2.*...
455             (ThermalConductivity(1:end-2,2:end-1)+...
456             ThermalConductivity(3:end,2:end-1)+...
457             ThermalConductivity(2:end-1,1:end-2)+...
458             ThermalConductivity(2:end-1,3:end)+...
459             4*ThermalConductivity(2:end-1,2:end-1)...
460             ) - ...
461             TNp1(1:end-2,2:end-1)/2.*...
462             (ThermalConductivity(1:end-2,2:end-1)+...
463             ThermalConductivity(2:end-1,2:end-1))-...
464             TNp1(3:end,2:end-1)/2.*...
465             (ThermalConductivity(3:end,2:end-1)+...
466             ThermalConductivity(2:end-1,2:end-1))-...
467             TNp1(2:end-1,1:end-2)/2.*...
468             (ThermalConductivity(2:end-1,1:end-2)+...
469             ThermalConductivity(2:end-1,2:end-1))-...
470             TNp1(2:end-1,3:end)/2.*...
471             (ThermalConductivity(2:end-1,3:end)+...
472             ThermalConductivity(2:end-1,2:end-1))+...
473             DeltaX^2*(...
474             1/DeltaT*(...
475             HeatCapacity(2:end-1,2:end-1).*...
476             (TNp1(2:end-1,2:end-1)-TN(2:end-1,2:end-1))...
477             -RhoWater*HeatVaporization(2:end-1,2:end-1).*...
478             (ThetaNp1(2:end-1,2:end-1)-ThetaN(2:end-1,2:end-1))...
479             )+ConvectiveCooling*(TNp1(2:end-1,2:end-1)-AmbientTemp)...
480             );
481
482
483         ErrWater(2:end-1,2:end-1)=WaterVaporPpNp1(2:end-1,2:end-1)/2.*...
484             (VaporConductivity(1:end-2,2:end-1)+...
485             VaporConductivity(3:end,2:end-1)+...
486             VaporConductivity(2:end-1,1:end-2)+...
487             VaporConductivity(2:end-1,3:end)+...
488             4*VaporConductivity(2:end-1,2:end-1)...
489             ) - ...
490             WaterVaporPpNp1(1:end-2,2:end-1)/2.*...
491             (VaporConductivity(1:end-2,2:end-1)+...
492             VaporConductivity(2:end-1,2:end-1))-...
493             WaterVaporPpNp1(3:end,2:end-1)/2.*...
494             (VaporConductivity(3:end,2:end-1)+...
495             VaporConductivity(2:end-1,2:end-1))-...
496             WaterVaporPpNp1(2:end-1,1:end-2)/2.*...
497             (VaporConductivity(2:end-1,1:end-2)+...
498             VaporConductivity(2:end-1,2:end-1))-...
499             WaterVaporPpNp1(2:end-1,3:end)/2.*...
500             (VaporConductivity(2:end-1,3:end)+...
501             VaporConductivity(2:end-1,2:end-1))+...
502             DeltaX^2*(...
503             RhoWater/DeltaT*(...
504             ThetaNp1(2:end-1,2:end-1)-ThetaN(2:end-1,2:end-1)...
```

```
505             )+ConvectiveVaporLoss*WaterMolecularWeight*...
506             (Porosity(2:end-1,2:end-1)-ThetaNp1(2:end-1,2:end-1))./...
507             Steffan(2:end-1,2:end-1).*...
508             (WaterVaporPpNp1(2:end-1,2:end-1)./...
509             (GasConstant*TNp1(2:end-1,2:end-1))-...
510             AmbientVaporPp/(GasConstant*AmbientTemp))...
511             );
512
513         ErrVector=[reshape(ErrHeat,NumCells,1);reshape(ErrWater,NumCells,1)];
514         ErrVector=sparse(ErrVector);
515         ErrVector=ErrVector(SolvingCellsDoubled);
516
517     end
518
519 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
520
521     function []=Compute_Partials()
522
523         dErrHeat_dT=zeros(NumRows,NumCols);
524         dErrHeat_dMatric=zeros(NumRows,NumCols);
525         dErrWater_dT=zeros(NumRows,NumCols);
526         dErrWater_dMatric=zeros(NumRows,NumCols);
527
528         dTheta_dMatric=-Theta1/LogNegMatric0*MatricNp1.^(-1);
529         dhumidity_dT=-humidity.*(WaterMolecularWeight/GasConstant*...
530             MatricNp1.*TNp1.^(-2));
531         dP_dT=dhumidity_dT.*PSat+humidity.*dPSat_dT;
532         dP_dMatric=WaterMolecularWeight/GasConstant*PSat.*humidity.*TNp1.^(-1);
533
534         try
535             dP_dT(OverPressure)=0;
536         end
537
538         try
539             dP_dMatric(OverPressure)=0;
540         end
541
542         dErrHeat_dMatric(2:end-1,2:end-1)=DeltaX^2/DeltaT*RhoWater*( ...
543             cWater*(TNp1(2:end-1,2:end-1)-TN(2:end-1,2:end-1))...
544             -HeatVaporization(2:end-1,2:end-1)).*dTheta_dMatric(2:end-1,2:end-1);
545
546         dErrHeat_dT(2:end-1,2:end-1)= ...
547             1/2*(ThermalConductivity(1:end-2,2:end-1)+...
548             ThermalConductivity(3:end,2:end-1)+...
549             ThermalConductivity(2:end-1,1:end-2)+...
550             ThermalConductivity(2:end-1,3:end)+...
551             4*ThermalConductivity(2:end-1,2:end-1)...
552             )+...
553             DeltaX^2*((HeatCapacity(2:end-1,2:end-1)-...
554             RhoWater*dHeatVaporization_dT*...
555             (ThetaNp1(2:end-1,2:end-1)-ThetaN(2:end-1,2:end-1)))/DeltaT + ...
556             ConvectiveCooling);
557
558         VaporConductivitySum=(VaporConductivity(1:end-2,2:end-1)+...
559             VaporConductivity(3:end,2:end-1)+...
560             VaporConductivity(2:end-1,1:end-2)+...
561             VaporConductivity(2:end-1,3:end)+...
562             4*VaporConductivity(2:end-1,2:end-1)...
563             )/2;
564
565         dErrWater_dMatric(2:end-1,2:end-1)= VaporConductivitySum .*...
566             dP_dMatric(2:end-1,2:end-1) + RhoWater*DeltaX^2/DeltaT *...
567             dTheta_dMatric(2:end-1,2:end-1) + ...
```



```
568             VaporLossCoef*(-dTheta_dMatric(2:end-1,2:end-1)./Steffan(2:end-1,2:end-1) .* ...
569             (WaterVaporPpNp1(2:end-1,2:end-1)./TNp1(2:end-1,2:end-1) - ...
570             AmbientVaporPp/AmbientTemp) + ...
571             (Porosity(2:end-1,2:end-1)-ThetaNp1(2:end-1,2:end-1))./Steffan(2:end-1,2:end-1) .* ...
572             dP_dMatric(2:end-1,2:end-1) .* (...
573             (WaterVaporPpNp1(2:end-1,2:end-1)./TNp1(2:end-1,2:end-1)-AmbientVaporPp/AmbientTemp) ./
574 ...
575             (AmbientVaporPp * Steffan(2:end-1,2:end-1)) + ...
576             TNp1(2:end-1,2:end-1).^(-1)));
577
578         dErrWater_dT(2:end-1,2:end-1)=VaporConductivitySum.*...
579             dP_dT(2:end-1,2:end-1)+...
580             VaporLossCoef * ...
581             (Porosity(2:end-1,2:end-1)-ThetaNp1(2:end-1,2:end-1))./Steffan(2:end-1,2:end-1) .* ...
582             (dP_dT(2:end-1,2:end-1)./(AmbientVaporPp * Steffan(2:end-1,2:end-1)) .* ...
583             (WaterVaporPpNp1(2:end-1,2:end-1)./TNp1(2:end-1,2:end-1)-AmbientVaporPp/AmbientTemp) -
584 ...
585             WaterVaporPpNp1(2:end-1,2:end-1).*TNp1(2:end-1,2:end-1).^(-2) + ...
586             dP_dT(2:end-1,2:end-1)./TNp1(2:end-1,2:end-1));
587
588         %Create Submatrix of partials Heat Error to T
589         dErrHeat_dT_im1j=zeros(NumRows,NumCols);
590         dErrHeat_dT_im1j(2:end-1,2:end-1)=-(ThermalConductivity(1:end-2,2:end-1)+...
591             ThermalConductivity(2:end-1,2:end-1))/2;
592
593         dErrHeat_dT_ip1j=zeros(NumRows,NumCols);
594         dErrHeat_dT_ip1j(2:end-1,2:end-1)=-(ThermalConductivity(2:end-1,2:end-1)+...
595             ThermalConductivity(1:end-2,2:end-1))/2;
596
597         dErrHeat_dT_ijm1=zeros(NumRows,NumCols);
598         dErrHeat_dT_ijm1(2:end-1,2:end-1)=-(ThermalConductivity(2:end-1,1:end-2)+...
599             ThermalConductivity(2:end-1,2:end-1))/2;
600
601         dErrHeat_dT_ijp1=zeros(NumRows,NumCols);
602         dErrHeat_dT_ijp1(2:end-1,2:end-1)=-(ThermalConductivity(2:end-1,3:end)+...
603             ThermalConductivity(2:end-1,2:end-1))/2;
604
605         diag_entries=[reshape(dErrHeat_dT_ijm1,NumCells,1),...
606             reshape(dErrHeat_dT_im1j,NumCells,1),...
607             reshape(dErrHeat_dT,NumCells,1),...
608             reshape(dErrHeat_dT_ip1j,NumCells,1),...
609             reshape(dErrHeat_dT_ijp1,NumCells,1)];
610
611         PM_HT=spdiags(diag_entries,[-NumRows,-1,0,1,NumRows],NumCells,NumCells);
612
613
614         %Create Submatrix of partials Water to Matric
615         dErrWater_dMatric_im1j=zeros(NumRows,NumCols);
616         dErrWater_dMatric_im1j(2:end-1,2:end-1)=-(VaporConductivity(1:end-2,2:end-1)+...
617             VaporConductivity(2:end-1,2:end-1))/2;
618
619         dErrWater_dMatric_ip1j=zeros(NumRows,NumCols);
620         dErrWater_dMatric_ip1j(2:end-1,2:end-1)=-(VaporConductivity(2:end-1,2:end-1)+...
621             VaporConductivity(1:end-2,2:end-1))/2;
622
623         dErrWater_dMatric_ijm1=zeros(NumRows,NumCols);
624         dErrWater_dMatric_ijm1(2:end-1,2:end-1)=-(VaporConductivity(2:end-1,1:end-2)+...
625             VaporConductivity(2:end-1,2:end-1))/2;
626
627         dErrWater_dMatric_ijp1=zeros(NumRows,NumCols);
628         dErrWater_dMatric_ijp1(2:end-1,2:end-1)=-(VaporConductivity(2:end-1,3:end)+...
629             VaporConductivity(2:end-1,2:end-1))/2;
630
```

```
631         diag_entries=[reshape(dErrWater_dMatric_ijm1,NumCells,1),...
632             reshape(dErrWater_dMatric_im1j,NumCells,1),...
633             reshape(dErrWater_dMatric,NumCells,1),...
634             reshape(dErrWater_dMatric_ip1j,NumCells,1),...
635             reshape(dErrWater_dMatric_ijp1,NumCells,1)];
636
637         PM_WM=spdiags(diag_entries,[-NumRows,-1,0,1,NumRows],NumCells,NumCells);
638
639         %Make submatricies of heat to matric and water to T.
640         PM_WT=spdiags(reshape(dErrWater_dT,NumCells,1),0,NumCells,NumCells);
641         PM_HM=spdiags(reshape(dErrHeat_dMatric,NumCells,1),0,NumCells,NumCells);
642
643
644         PartialsMatrix=[PM_HT,PM_HM;PM_WT,PM_WM];
645
646         PartialsMatrix=PartialsMatrix(SolvingCellsDoubled,:);
647         PartialsMatrix=PartialsMatrix(:,SolvingCellsDoubled);
648
649
650
651     end
652
653 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
654
655     function []=UpdateVapor()
656
657         PhiAir=1-ThetaNp1-PhiMineral-PhiOrganic;
658
659         HeatCapacity=ThetaNp1*RhoWater*cWater + PhiMineral*RhoMineral*cMineral...
660             + PhiOrganic*RhoOrganic*cOrganic; %de Vries 1996 eqn (7.1) J/(kg m^3)
661
662         HeatVaporization=(45144-48*(TNp1-StandardTemp))/WaterMolecularWeight;
663         %Campbell 1994 pg. 309 (J/kg)
664
665         %Relative Humidity
666         humidity=exp(WaterMolecularWeight*MatricNp1./(GasConstant*TNp1));
667         %Campbel 1995 eqn (8)
668
669         S=1-373.15*TNp1.^(-1); %Used in Saturation Vapor Pressure calculations
670
671         %Saturation Vapor Pressure of water (Pa) Campbell 1995 eqn(10)
672         PSat=101325*exp(13.3016*S-2.042*S.^2+0.26*S.^3+2.69*S.^4);
673
674         %Derivative Saturation Vapor Pressure w.r.t. T (Pa/K)
675         dPSat_dT=373.15*PSat.*(13.3016-4.084*S+0.78*S.^2+10.76*S.^3)./TNp1.^2;
676
677         %Water Vapor Partial Pressure (Pa)
678         WaterVaporPpNp1=humidity.*PSat;
679
680         %My fix
681         OverPressure=WaterVaporPpNp1>0.9*AmbientPressure;
682
683         try
684             WaterVaporPpNp1(OverPressure)=0.9*AmbientPressure;
685         end
686
687         Steffan=1-WaterVaporPpNp1/AmbientPressure;
688
689         %Campbell's Fix see 1995 pg.
690         %if sum(sum(Steffan>3.33))>1;
691         %    Steffan(Steffan>3.33)=3.33;
692         %end
693     end
```



```
694
695 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
696
697     function []=UpdateConductivities()
698
699
700         %Diffusivity of water in air (m^2/hr) Campbell 1995 eqn (13)
701         D_v=Dv0*(StandardPressure/(AmbientPressure*StandardTemp^1.75))*(TNp1).^(1.75);
702
703
704         %Vapor Conductivity of Duff (m^2 Pa hr) Campbell 1995 eqn. (12)
705         VaporConductivity=Xi*Eta*WaterMolecularWeight*D_v.*(Porosity-ThetaNp1)...
706             ./(GasConstant*TNp1.*Steffan);
707
708         boiling_indices=TNp1>373.15;
709
710         %Empirical Soil Parameter, Campbell 1994 eqn (4)
711         q=(TNp1/303).^4; %4th power in Campbell's code 2nd in paper
712         q(boiling_indices)=2.3; %Following Campbell's code
713         q=Q0*q;
714
715         %Weighting function for lambda_f Campbell 1994 eqn (3)
716         indices = ThetaNp1 < 0.01*Theta0;
717         f_w=(1+(ThetaNp1/Theta0).^(-q)).^(-1);
718         f_w(indices)=0; %Follows Campbell 1995 code
719
720         %Thermal Conductivity of Dry Air (J/(m hr K)) Campbell eqn. (9)
721         lambda_DryAir=(0.024+7.73e-5*(TNp1-StandardTemp)-...
722             2.6e-8*(TNp1-StandardTemp).^2)*3600;
723
724
725         %Thermal Conductivity of Water (J/(m hr K)) from Tarnawski et al.
726         %2000 eqn (A4).
727         lambda_w=(0.569+1.884e-3*(TNp1-StandardTemp)-0.0772e-5*(TNp1-StandardTemp).^2)*3600;
728
729         lambda_w(boiling_indices)=8280; %2.3*3600 - Following Campbell Code
730
731         %Thermal Conductivity of air (J/(m hr K))
732         lambda_a=lambda_DryAir + ...
733             (WaterMolecularWeight*HeatVaporization.*f_w.*D_v)./ ...
734             (GasConstant*Steffan.*TNp1).*dPSat_dT;
735
736         %Fluid Conductivity (J/(m hr K)) Campbell 1994 eqn (2)
737         lambda_f=lambda_a + f_w .* (lambda_w-lambda_a);
738
739         %Weighting Factor for water
740         k_w=1/3*(2*(1+(lambda_w./lambda_f-1)*ShapeFactorOrganic).^(-1)+ ...
741             (1+(lambda_w./lambda_f-1)*(1-2*ShapeFactorOrganic)).^(-1));
742
743         %Weighting Factor for air
744         k_a=1/3*(2*(1+(lambda_a./lambda_f-1)*ShapeFactorOrganic).^(-1)+ ...
745             (1+(lambda_a./lambda_f-1)*(1-2*ShapeFactorOrganic)).^(-1));
746
747         %Weighting Factor for mineral
748         k_m=1/3*(2*(1+(lambda_m./lambda_f-1)*ShapeFactorMineral).^(-1)+ ...
749             (1+(lambda_m./lambda_f-1)*(1-2*ShapeFactorMineral)).^(-1));
750
751         %Weighting Factor for organic
752         k_o=1/3*(2*(1+(lambda_o./lambda_f-1)*ShapeFactorOrganic).^(-1)+ ...
753             (1+(lambda_o./lambda_f-1)*(1-2*ShapeFactorOrganic)).^(-1));
754
755         %Thermal Conductivity of soil (J/(m hr K))
756         ThermalConductivity=(k_w.*ThetaNp1.*lambda_w + k_a.*PhiAir.*lambda_a ...
```

```
757             + k_m.*PhiMineral.*lambda_m + k_o.*PhiOrganic.*lambda_o)./ ...
758             (k_w.*ThetaNp1 + k_a.*PhiAir + k_m.*PhiMineral + k_o.*PhiOrganic);
759
760         ThermalConductivity(ThermalConductivity<0 | ThermalConductivity> 18000)=3600;
761         %Following Campbell Code.
762
763     end
764
765 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
766
767     function []=UpdateBeta()
768
769         %Determine Locations where neighbors have source.
770         Source=[BetaN(2:end,:)==1;zeros(1,NumCols)] + ...
771             [zeros(1,NumCols);BetaN(1:end-1,:)==1] + ...
772             [BetaN(:,2:end)==1,zeros(NumRows,1)] + ...
773             [zeros(NumRows,1),BetaN(:,1:end-1)==1];
774
775
776         %FrandsenProb = Chi ./ (1 + exp(-(B0 + B1*Gamma + B2*Alpha + B3*Rho)));
777         %FrandsenProb = 1 ./ (1 + exp(-(B0 + B2*Alpha + B3*Rho)));
778         %FrandsenProb(Gamma>0.1)=0;
779         FrandsenProb = (1 + exp(-(B0 + B1*Gamma + B2*Alpha + B3*Rho))).^(-1);
780
781         BetaNp1=BetaN;
782
783         BurnTimeSteps(BurningCells)=BurnTimeSteps(BurningCells)-1;
784         BetaNp1(BurnTimeSteps==1)=1;
785
786         BetaNp1(BetaN==1)=2; %At boundry only one time step
787
788         BetaNp1(BetaN==0 & Source > 0 & rand(NumRows,NumCols)<FrandsenProb)=0.9;
789
790     end
791
792 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
793
794     function []=SetSolvingCells()
795
796         BurningCellNumbers=find(BurningCells==1); %Locate Burning Cells
797
798         BurningCellNumbers=[(1:NumRows)';BurningCellNumbers;...
799            (NumCells-NumRows+1:NumCells)';...
800             NumRows*(2:NumCols-1)';NumRows*(1:NumCols-2)'+1]; %Add boundaries
801
802         SolvingCells=setdiff((1:NumCells)',BurningCellNumbers);
803         %Solve everwhere except on boundary and burning cells.
804
805         NumSolvingCells=length(SolvingCells);
806         SolvingCellsDoubled=[SolvingCells;SolvingCells+NumCells];
807
808     end
809
810 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
811
812     function []=make_plots()
813         pause on
814
815         subplot(2,2,1)
816         pcolor(BetaN(2:end,2:end))
817         colorbar
818         title('\beta')
819
```



```
820         subplot(2,2,2)
821         pcolor(TN(2:end,2:end)-StandardTemp)
822         colorbar
823         title('T - Celcius')
824
825         subplot(2,2,3)
826         pcolor(Gamma(2:end,2:end))
827         colorbar
828         title('\gamma')
829
830 %         subplot(2,2,4)
831 %         pcolor(sign(WaterVaporPpNp1(2:end,2:end)./TN(2:end,2:end)-AmbientVaporPp/AmbientTemp).*...
832 %             log(abs(WaterVaporPpNp1(2:end,2:end)./TN(2:end,2:end)-AmbientVaporPp/AmbientTemp)+1))
833 %         colorbar
834 %         title('VaporPressure')
835
836         subplot(2,2,4)
837         pcolor(FrandsenProb(2:end,2:end))
838         colorbar
839         title('Frandsen Prob.')
840
841         pause(0.01)
842
843     end
844
845 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
846
847     function [GammaLocal]=VMC2GMC(ThetaLocal)
848        GammaLocal=RhoWater.*ThetaLocal./Delta;
849     end
850
851 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
852
853     function [ThetaLocal]=GMC2VMC(GammaLocal)
854        ThetaLocal=Delta.*GammaLocal./RhoWater;
855     end
856
857 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
858
859     function [Matric]=VMC2Matric(Theta)
860         %Campbell 1995 eqn. (9) inverted
861         Matric=-(-Matric0).^(1-Theta/Theta1);
862     end
863
864 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
865
866     function [Theta]=Matric2VMC(Matric)
867         %Campbell 1995 eqn. (9)
868         Theta=Theta1*(1-log(-Matric)/LogNegMatric0);
869     end
870
871 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
872
873
874 end
875
876
```