Supplementary material
Thermal non-equilibrium of porous flow in a resting matrix applicable to melt migration: a parametric study

Laure Chevalier[1,2], Harro Schmeling[1]

[1]Institute of Geosciences, Goethe University, 60438 Frankfurt, Germany
[2]Laboratory example, city, postal code, country

Correspondence to: Harro Schmeling (schmeling@geophysik.uni-frankfurt.de)

1 Analytical solution

Eq. (19) (in main paper) is a second order differential equation for $T_f - T_s$ with constant parameters. Its solution is composed of the sum of a particular solution and of the solution of the corresponding homogeneous differential equation.

1.1 Particular solution

The right hand side of Eq. (19) is a constant. In this special case with constant parameters, a simple way to find a particular solution consists in assuming that for this solution, $T_f - T_s$ is a constant. Its derivative is equal to zero, and the constant value of $T_f - T_s$ is:

$$T_f - T_s = (1-\phi)\frac{Pe}{A}\frac{\Delta T}{H} \quad \text{(S1)}$$

1.2 Homogeneous solution

We now need to find the general solution of the homogeneous equation

$$\frac{1}{A}\frac{\partial^2(T_f - T_s)}{\partial z^2} - \frac{Pe}{A}\frac{\partial(T_f - T_s)}{\partial z} - \frac{1}{1-\phi}\left(T_f - T_s\right) = 0 \quad \text{(S2)}$$

The second order algebraic equation associated with Eq. (S2) is

$$r^2 - Pe\, r - \frac{A}{1-\phi} = 0 \quad \text{(S3)}$$

Its determinant and roots are

$$\Delta = Pe^2 + \frac{4A}{1-\phi} \quad , \; r_1 = \frac{Pe - \sqrt{Pe^2 + \frac{4A}{1-\phi}}}{2}, \; r_2 = \frac{Pe + \sqrt{Pe^2 + \frac{4A}{1-\phi}}}{2}$$

and the solution of Eq. (S2) is of the form:

$$T_f - T_s = \alpha e^{r_1 z} + \beta e^{r_2 z} + (1-\phi)\frac{Pe}{A}\frac{\Delta T}{H} \quad \text{(S4)}$$

We now need to determine α and β from boundary conditions.

1.3 Constraints from boundary conditions

At z = 0, both T_f and T_s values are set constant, to the same value. Therefore we have the condition $T_f - T_s = 0$ at z = 0. We then have the following relationship for α and β:

$$\alpha + \beta = -(1-\phi)\frac{Pe}{A}\frac{\Delta T}{H} \tag{S5}$$

Besides, at $z = H$, both T_f and T_s gradients are constant and equal to each other. Thus we have the condition $\frac{\partial(T_f - T_s)}{\partial z} = 0$ at $z = H$. This gives the following relationship for α and β :

$$\alpha r_1 e^{r_1 H} + \beta r_2 e^{r_2 H} = 0 \tag{S6}$$

From Eq. (S5) and (S6) we get

$$\alpha = (1-\phi)\frac{Pe}{A}\frac{\Delta T}{H}\frac{r_2}{r_1 e^{(r_1 - r_2)H} - r_2}, \quad \beta = (1-\phi)\frac{Pe}{A}\frac{\Delta T}{H}\frac{r_1}{r_2 e^{(r_2 - r_1)H} - r_1} \tag{S7}$$

1.4 Comparison with numerical models

Here a Figure is shown (Fig. S1) in which the analytical solutions (Eq. 20) of the simplified ordinary differential equation (19) are compared to the numerical time-dependent models.

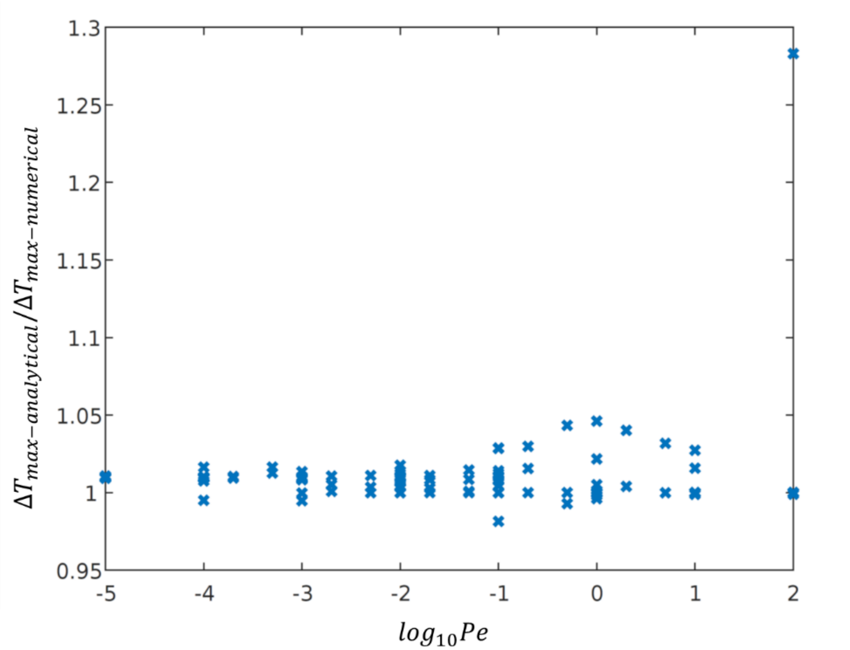

Figure S1. Ratio of maximum temperature differences fluid – solid from the simplified analytical solution equ. (20) to the numerically determined maximum temperature differences for all 123 models.

2 Limits determination

2.1 Limit A → 0

Figure 2a and b represent T_f and T_s as functions of z at different times for two different models. In both models, Pe = 1, A = 1, To derive Eq. (23) we expand the quantities r_1, r_2, α, β given by Eq. (21) and (22) into Taylor series in terms of A around A = 0:

$$r_1 = -\frac{A}{Pe\,(1-\phi)} + O(A^2), \quad r_2 = Pe + \frac{A}{Pe\,(1-\phi)} + O(A^2) \tag{S8}$$

$$\alpha = -(1-\phi)\frac{Pe}{A}\frac{\Delta T}{H}\left(1 - \frac{Ae^{-Pe\,H}}{Pe^2\,(1-\phi)}\right) + O(A^2), \qquad \beta = -\frac{\Delta T}{H}\frac{1}{Pe\,(1-\phi)}(1 + O(A)) \tag{S9}$$

Inserting these terms into Eq. (20) results in

$$T_f - T_s = \frac{\Delta T}{H}\left(z + \frac{1}{Pe e^{PeH}}(1 - e^{Pez})\right) + O(A) \tag{S10}$$

which, in the limit of A → 0, is equal to Eq. (23). Figure S2 presents a comparison of results from Eq. (20) with the different limits we derive. Results from Eq. (23, i.e. S10) at $z = H$ are in good agreement with Eq. (20) (Fig.S2b) for $A < 10^{-2}$ (having Pe = 1 and ϕ = 0.1), except for very small Pe (Fig. S2c). This is expected considering that A is no more negligible with respect to Pe (see below)

2.2 Limit $A \to \infty$

The limit Eq. (24) is derived by straightforward applying limit rules for the case of $A \to \infty$ to equations (20) – (22). Figure S2b shows that when choosing Pe =1 and ϕ=0.1 this limit is in good agreement with Eq. (20) for A > 1.

2.3 Limit $Pe \to 0$

When Pe tends to 0, it becomes negligible with respect to $\sqrt{\frac{4A}{1-\phi}}$. We then get

$$r_1 \to -\sqrt{\frac{A}{1-\phi}}, \qquad r_2 \to +\sqrt{\frac{A}{1-\phi}}$$

and the following limit for $(T_f - T_s)$ and definition of a function M

$$T_f - T_s = (1-\phi)\frac{Pe}{A}\frac{\Delta T}{H}\left(1 - \frac{e^{-\sqrt{\frac{A}{(1-\phi)}}z}}{1+e^{-\sqrt{\frac{A}{(1-\phi)}}2H}} - \frac{e^{\sqrt{\frac{A}{(1-\phi)}}z}}{1+e^{\sqrt{\frac{A}{(1-\phi)}}2H}}\right) \equiv (1-\phi)\frac{Pe}{A}\frac{\Delta T}{H}(1-M) \tag{S11}$$

We now consider the case in which Pe tends to 0, and A is also very small (Pe and A tend to 0 but Pe/A <<1). This is why starting from low A limit doesn't work for getting this limit. In this case, we can look at the limit of Eq. (S11) when A tends to

0. For more readability, in eq. (S11) we call M the terms with the exponentials in the parenthesis, and $N = \sqrt{\frac{A}{1-\phi}}$. We first rearrange the term M in:

$$M = -\frac{e^{-\sqrt{N}\,z} + e^{\sqrt{N}\,z} + e^{-\sqrt{N}\,(2H-z)} + e^{\sqrt{N}\,(2H-z)}}{2 + e^{-2\sqrt{N}\,H} + e^{2\sqrt{N}\,H}} \tag{S12}$$

When x tends to 0, we can use the following limit for exponentials from Taylor series developments taking into account order 2 terms:

$$e^{-x} + e^{x} = 1 - x + \frac{x^2}{2} - \cdots + 1 + x + \frac{x^2}{2} + \cdots = 2 + x^2 + O(x^4) \tag{S13}$$

Applying this to Eq. (S12) we get

$$M \to -\frac{4 + Nz^2 + N(4H^2 - 4Hz + z^2)}{4 + 4NH^2} = -1 + \frac{NH\left(z\left(1-\frac{z}{2H}\right)\right)}{1 + NH^2} \tag{S14}$$

We now re-insert M in Eq. (S11), and since N tends to 0 we consider that NH^2 is negligible with respect to 1. We then get the following limit for $(T_f - T_s)$:

$$T_f - T_s = Pe\Delta T z\left(1 - \frac{z}{2H}\right) \tag{S15}$$

Fig. S2c presents the values predicted by eq. (20), (23), (25) and (26) when A tends to 0, for the case of a very small Pe value ($Pe = 10^{-9}$). One can see that in this case eq. (25) and (26) give better fits than Eq. (23), which is reasonable since $Pe << A$.

2.4 Limit $Pe \to \infty$

To obtain the limit of Eq. (20) for $Pe \to \infty$ also allowing for finite ratios Pe/A we write Eq. (20) in terms of $C = 4A/((1-\phi)Pe^2)$ and determine the limit for $C \to 0$. The terms α, β in Eq. (20) can be linearized with respect to *C*. Inserting them into Eq. (20) gives

$$T_f - T_s = \frac{(1-\phi)\Delta T}{H}\frac{Pe}{A}\left[-\left(1 - \frac{1}{4}C\, e^{-Pe\left(1+\frac{1}{2}C\right)H}\right) e^{-Pe\frac{1}{4}Cz} + \left(-\frac{1}{4}C\left(1 - \frac{1}{4}C\right) e^{-Pe\left(1+\frac{1}{2}C\right)H}\right) e^{Pe\left(1+\frac{1}{4}C\right)z} + 1\right] \tag{S16}$$

Allowing still for a finite term (*C Pe*), in the limit of $C \to 0$ Eq. (S16) turns into

$$T_f - T_s = \frac{(1-\phi)\Delta T}{H}\frac{Pe}{A}\frac{1 - e^{-\frac{Az}{(1-\phi)Pe}}}{A/Pe} \tag{S17}$$

Substituting x = 1/*Pe* and applying the rule of L'Hospital we get

$$\lim_{x\to 0}\left(T_f - T_s\right) = \lim_{x\to 0}\frac{(1-\phi)\Delta T}{H}\frac{-e^{-\frac{Az}{(1-\phi)}x}\left(-\frac{Az}{1-\phi}\right)}{A} = \frac{\Delta T}{H}z \tag{S18}$$

One can see in Fig. (S2a) that this limit predicts ΔT_{max} values in very good agreement with Eq. (20) for $Pe > 100$ (having A=1 and ϕ=0.1).

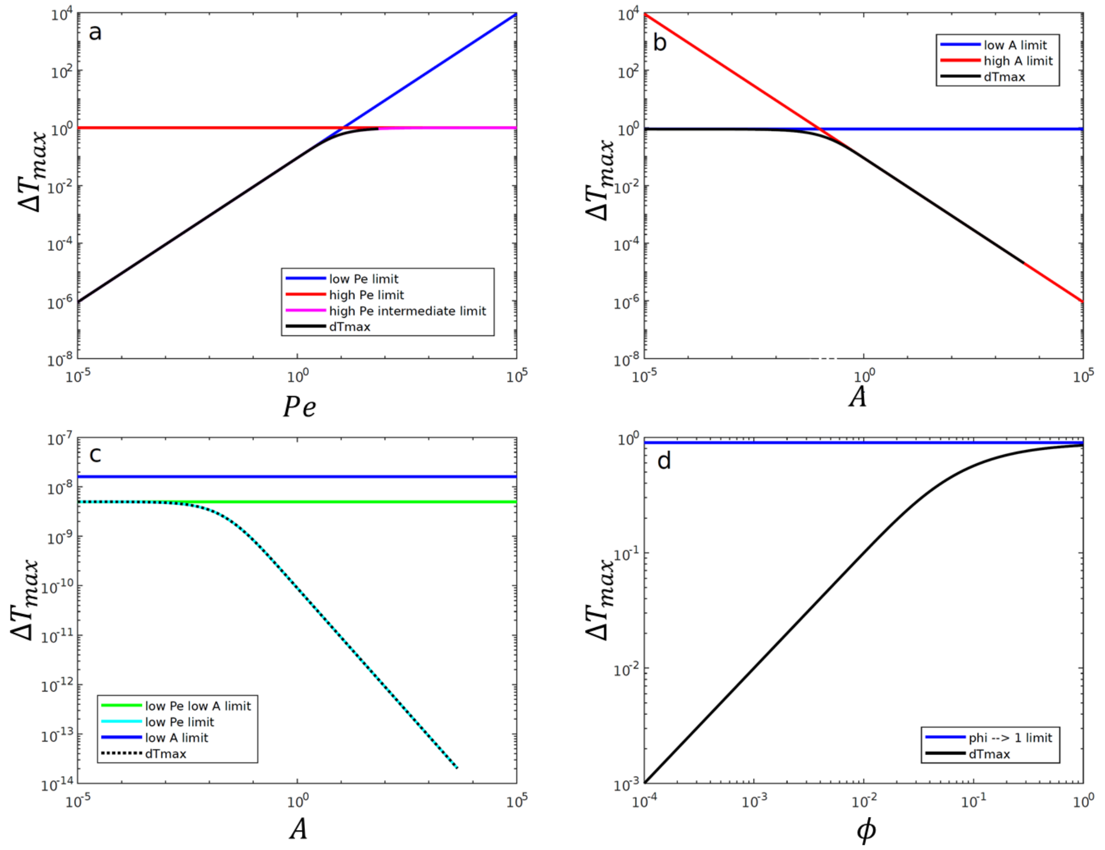

Figure S2. Comparison of the analytic solution equ. (20) with the different limits derived in section 4.3. The black curves represent the analytic solutions, the colored straight lines show the results in the high or low value limits of equ. (23) to (27), respectively. The used values in the different figures are a) A = 1, ϕ=0.1, b) Pe =1 and ϕ=0.1, c) Pe = 1 and ϕ = 0.1, d) A = 1 and Pe = 1.

3 Boundary conditions

To demonstrate the effect of different boundary conditions we show three models from different regimes, each calculate with three different boundary conditions (Fig. S3).

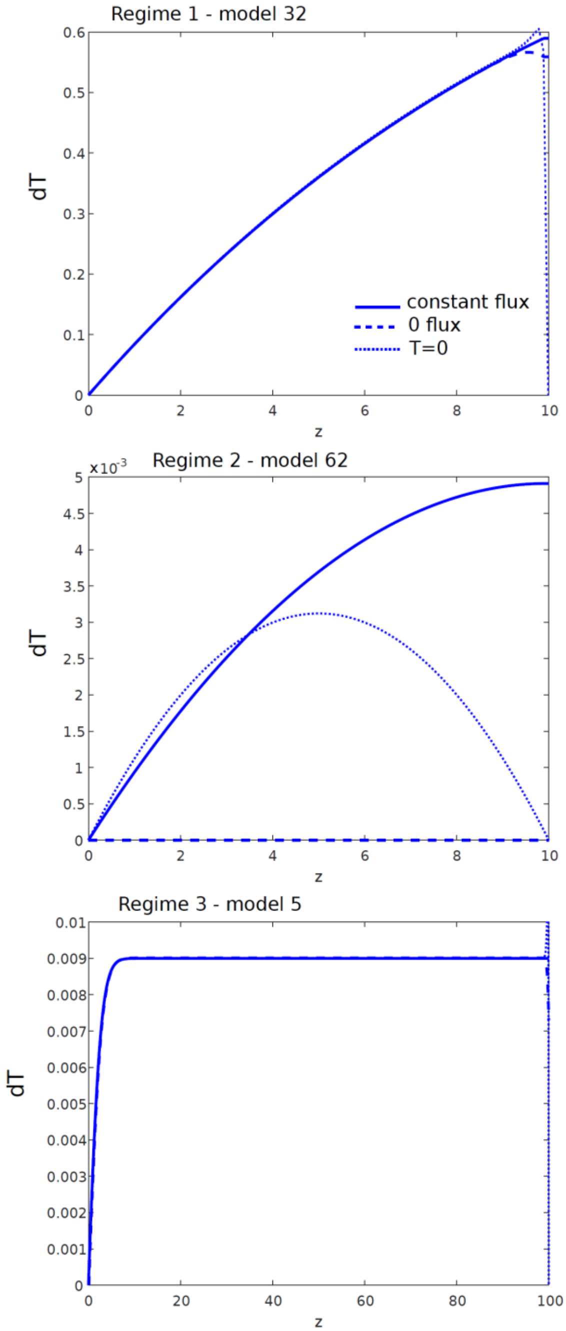

Figure S3. Vertical profiles of $(T_f - T_s)$ at late times of evolution for models representing different regimes. Regime 1: high Pe, regime 2: small A, regime 3: large A. For each regime three different top boundary conditions have been assumed as indicated in the legend.

4 Different material properties of solid and fluid

Here we show Figure S4 demonstrating the influence of various contrasts of thermal propertied between solid and fluid. In Figure S6 material properties typical for water in sedimentary rock is chosen and the effect of various *Pe, A, H* combinations are shown.

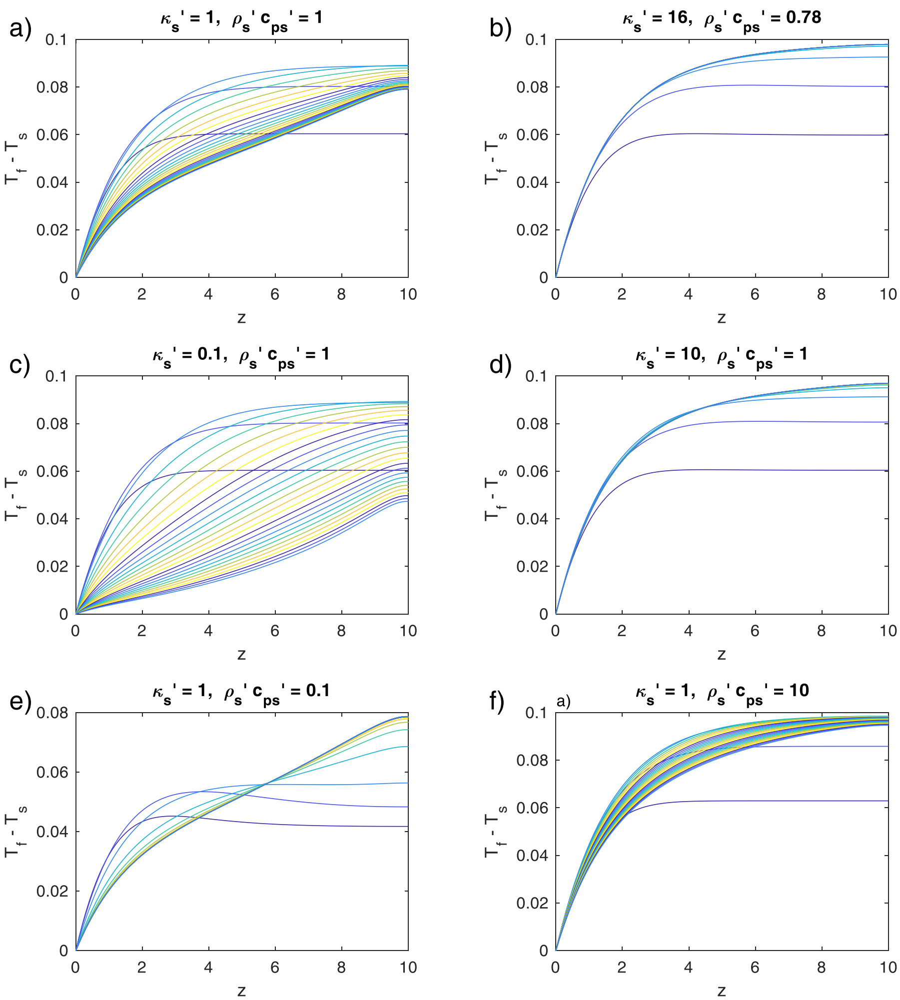

Figure S4. Time- and depth- dependent profiles of the fluid – solid temperature differences as in Fig.5. a) reference models (as in Fig. 5a) with Pe = 1, A = 1, H = 10 and equal fluid to solid properties. b) to f) profiles as in a) but with fluid to solid properties ratios as indicated in the sub-figure titles, Pe =1 and $A\lambda_{eff}' = 1$. The properties in b) are typical for water in sedimentary rocks.

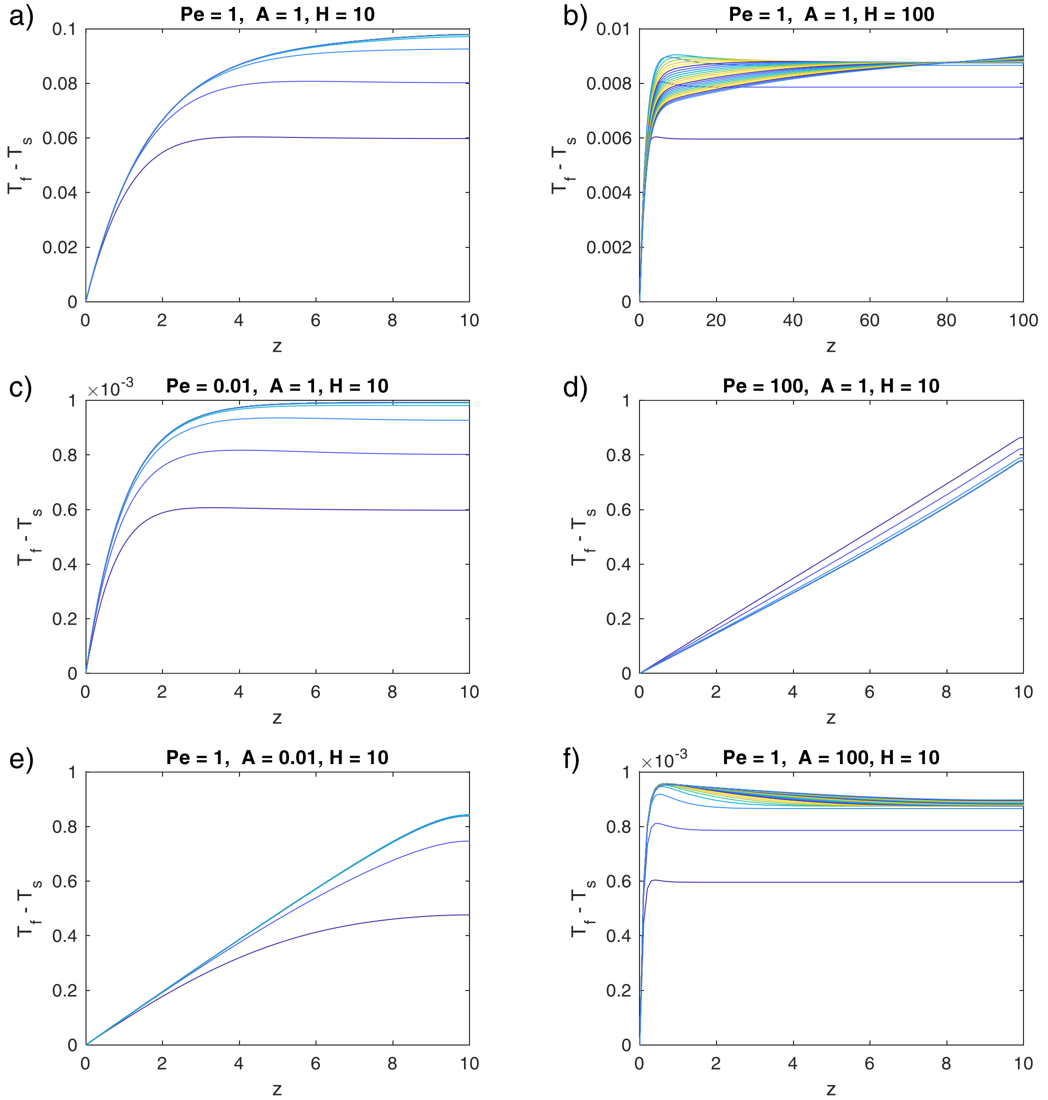

Figure S5. Time- and depth- dependent profiles of the fluid – solid temperature differences as in Fig. 5, but for fluid to solid property ratios typical for water flowing through sedimentary rocks, i.e. $\rho_s' c_{p,s}{}' = 0.78$, $\kappa_s' = 16$, $A\lambda_{eff}{}' = 1$. Pe, A, and H have been chosen as indicated in the sub-figure titles (as in Fig. 5).

5 Time-dependent A

Figure S6 shows the evolution of thermal non-equilibrium assuming a time-dependent heat transfer parameter *A*.

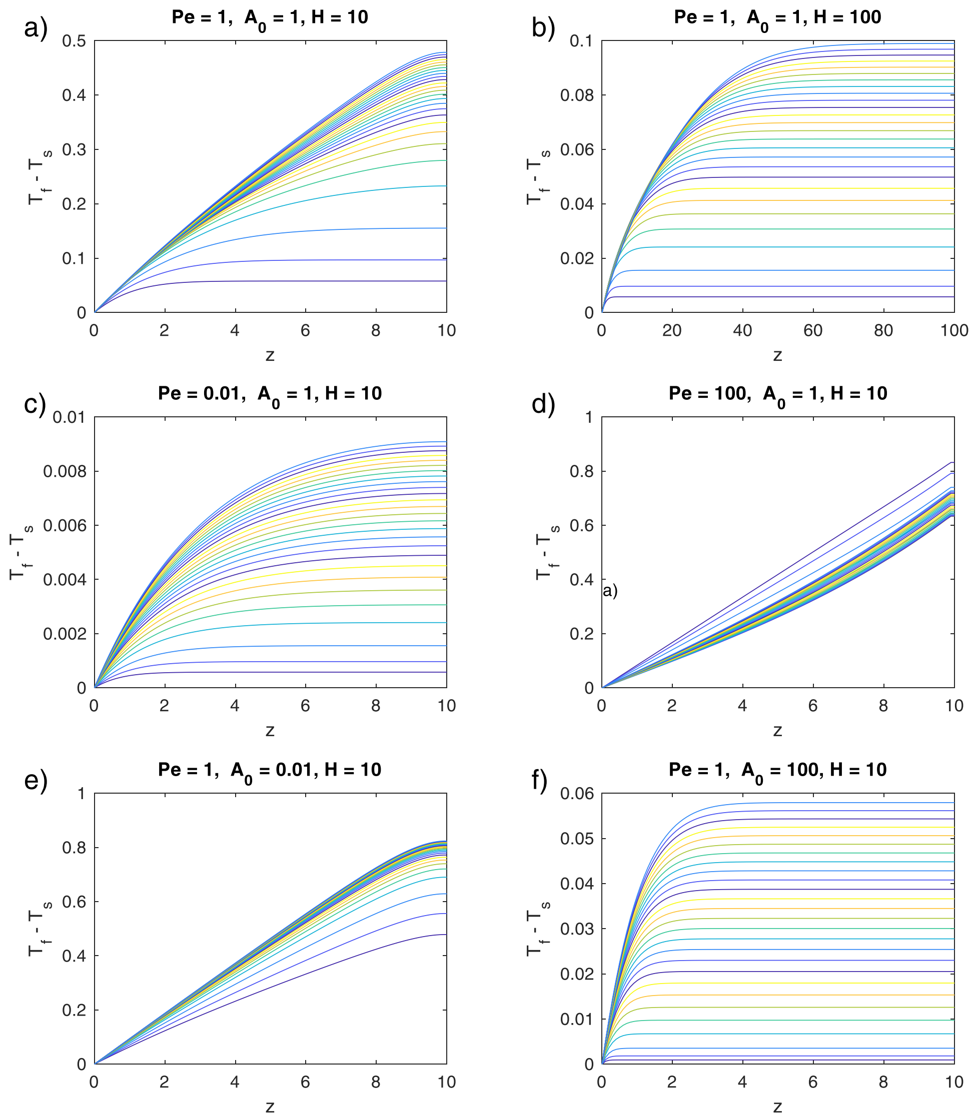

Figure S6. Time- and depth-dependent profiles of fluid – solid temperature differences for time dependent heat transfer parameter A according to thermal boundary layer theory (equ. 30). Else as Fig. 5.

6 Numerical programs

Here we give the Matlab routines used to calculate fluid to solid temperatures differences to generate Figures like Fig.5.

Main program:

```
% Parent mfile for runing parametric models. For every model it creates a
% new directory, in which key temperature results and outputfiles from the
% model solving are written. This code uses the function
% LTNEbasicdt. This function solves non-thermal equilibrium two-phase
% flow (static matrix) and returns the fluid, solid temperatures and the
% temperature difference at top at the end of the run, as well as the
% maximum temperature difference at top recorded during the system
% evolution. Input parameters are :
% (Pe,A,phi0,H,delT,modelname,dt,tmax,outputfactor)
% Pe: Péclet number
% A: Heat transfer number
% phi0: Porosity (fluid volume fraction)
% H: Height of the domain (normalised with fluid unit scale : width of
% dike for e.g.
% delT: Initial temperature difference between top and bottom
% modelname: Name of the model
% dt : Time step
% tmax : Ending time
% outputfactor : Basic outputs come every 2000 timesteps. this can be
% changed using outputfactor (n.b. the first outputs come at the 500th time
% step. If you want to reduce the time between two outputs, the choosen
% outputfactor must be a divisor of 500)

% Beside the LTNEbasicdt function parameters that must be defined for every
% model, the user should also check that the number of models to solve
% (nmodel) is correct. For every model a name must be given. e.g. here :
% model1. This name must be the same as the one in the LTNEbasicdt
% arguments. Also, the first column of "Results" must be manually entered.
% Here we entered "1" which references to "model1", so that associated
% values can be easily found in the final Results file. another option
% would be to have one result file for every model, which can be a good
% idea if you are not sure of the choosen dt, for example, since if matlab
% crashes at the 9th model, you won't get results from the 8 preceding.
% You could retrieve them from the function outputfiles though.

clearvars
close all

nmodel = 1; % number of models to be run

Results = zeros(nmodel,6);
```

```
mkdir('model1')
[Tftop,Tstop,dTtop,dTtopmax,kmax]=LTNEbasicdtpaper(1,1,0.1,1e1,1,'model1',1e-
3,100,2);
Results(1,:) = [1,Tftop,Tstop,dTtop,dTtopmax,kmax];
save(['1' '_' 'values' '.txt'],'Results','-ascii')
```

Routine which is called by the above program:

```
% Function LTNEbasicdt to be used with a parent mfile in which the model to
% be run is defined. This function solves non-thermal equilibrium two-phase
% flow (static matrix) and returns the fluid, solid temperatures and the
% temperature difference at top at the end of the run, as well as the
% maximum temperature difference at top recorded during the system
% evolution. Input parameters are :
% Pe: Péclet number
% A: Heat transfer number
% phi0: Porosity (fluid volume fraction)
% H: Height of the domain (normalised with fluid unit scale : width of
% dike for e.g.
% delT: Initial temperature difference between top and bottom
% modelname: Name of the model
% dt : Time step
% tmax : Ending time
% outputfactor : Basic outputs come every 2000 timesteps. this can be
% changed using outputfactor (n.b. the first outputs come at the 500th time
% step. If you want to reduce the time between two outputs, the choosen
% outputfactor must be a divisor of 500)


function
[Tftop,Tstop,dTtop,dTtopmax,kmax]=LTNEbasicdttest(Pe,A,phi0,H,delT,modelname,dt,tma
x,outputfactor)

ncolor=8;
cmap = parula(ncolor);
kcol =1;
dx = 1e-1; % Grid size
output = 1; %1=outputs, other : no output files

% Prepare other parameters
prefixe = strcat('./',modelname,'/',modelname);
x = 0:dx:H;
nx = length(x);
t    = dt:dt:tmax;

Tf = zeros(nx,1);
difTtop = [];

%Initial condition
```

```
for i = 1:nx
    Tf(i) = delT           -x(i)/H; % constant gradient
%     Tf(i) = 0;
end
Tm=Tf;
Tmnew =Tm;
Tfnew=Tf;

%Boundary conditions at start
Tm0 = delT;
Tf0 = delT;
Tm1 = 0;
Tf1 = 0;

Tmnew(1) = Tm0;
Tm(1)    = Tm0;

Tmnew(nx) = Tm1;
Tm(nx)    = Tm1;

Tfnew(1) = Tf0;
Tf(1) = Tf0;

Tfnew(nx) = Tf1;
Tf(nx) = Tf1;

%figure initial conditions
% figure;
%plot(x,Tfnew,'k',x,Tmnew,'k')
hold on

if output == 1
    Qsave = [x' Tfnew Tmnew];
    save([prefixe '_' num2str(0) '.txt'],'Qsave','-ascii')
end

dTtopmax=0;
for k =2:length(t)
    for i = 2:nx
        % FTCS with upwind
        if i <nx
            Tmnew(i)=Tm(i)+dt*((1/A)*(Tm(i+1)-2*Tm(i)+Tm(i-1))/dx^2+...
                (phi0/(1-phi0))*(Tf(i)-Tm(i)));
            Tfnew(i)= Tf(i) + dt*(-(Pe/A)*(Tf(i)-Tf(i-1))/dx +...
                (1/A)*(Tf(i+1)-2*Tf(i)+Tf(i-1))/dx^2  - (Tf(i)-Tm(i)));
        end
        if i == nx
            Tmnew(i) = Tmnew(i-1)-dx/H; %constant flux condition at top (Neumann)
            Tfnew(i) = Tfnew(i-1)-dx/H;
        end
        dTtopmaxlast = dTtopmax;
```

```
        dTtopmax = max(dTtopmax,Tfnew(nx)-Tmnew(nx));
        if dTtopmax > dTtopmaxlast ; kmax = k; end
    end

    if k < (1000*outputfactor+1)
        kmod = mod(k,500*outputfactor);
        if kmod == 0
%            plot(x,Tfnew,'r',x,Tmnew,'b')
            plot(x,Tfnew-Tmnew,'Color',cmap(kcol,:))
            kcol=kcol+1;
            if kcol>ncolor;kcol=1;end
            difTtop=[difTtop,Tfnew(nx)-Tmnew(nx)];
            if output ==1
                Qsave = [x' Tfnew Tmnew];
                save([prefixe '_' num2str(k*dt) '.txt'],'Qsave','-ascii')
            end
            t(k)
        end

    else
        kmod = mod(k,2000*outputfactor);
        if kmod == 0
%            plot(x,Tfnew,'r',x,Tmnew,'b')
%             plot(x,Tfnew-Tmnew,'b')
            plot(x,Tfnew-Tmnew,'Color',cmap(kcol,:))
                        kcol=kcol+1;
            if kcol>ncolor;kcol=1;end

           difTtop=[difTtop,Tfnew(nx)-Tmnew(nx)];
            if output == 1
                Qsave = [x' Tfnew Tmnew];
                save([prefixe '_' num2str(k*dt) '.txt'],'Qsave','-ascii')
            end
            t(k)

        end
     end


    drawnow

    Tf = Tfnew;
    Tm = Tmnew;
end

xlabel('z ')
ylabel('T_f-T_s')

if output == 1
    save([prefixe '_' 'outputdata.txt'],'x','Tfnew','Tmnew','-ascii')
    print('-f1','-dpng',prefixe)
```

```
end
% close (1)

%Key values returned from function
Tftop = Tfnew(nx);
Tstop = Tmnew(nx);
dTtop = Tfnew(nx)-Tmnew(nx);
%dTtopmax = max(difTtop);
box on
```

Main program for analytical solution:

```
%program for drawing dTmax as a function of Pe and A. Parameters : dT, H, z
%and phi. This program uses the analytical solution for dTmax, and some
%limits, where the analytical solution is not solvable by matlab (large
%exponential exponents for example). The domains where limits have to be
%used instead of the analytical solution must be precised by the user
%(lines marked with %%%%%%%%%). In this version, only the limits for high A
%and high Pe were used. Others can be added if needed (see functions in the
%same directory).

clearvars
% close all
hold on

dT = 1;
H = 10;
z = [0:0.01*H:H];
phi = 0.1;

Pe = 1;
A = 1;




for k = 1:length(z)

        dTmax(k) = dTmaxcalc(Pe,A,phi,dT,H,z(k));

%           [dTmax0(k),dTmax(k)] = dTmaxcalchighPe(Pe,A,phi,dT,H,z(k));

end


plot(z,dTmax,'r--','linewidth',2)
xlabel('z')
```

```
ylabel('dTmax')

box on
ylabel('T_f - T_s')
title('Pe = 1, A = 1, H = 10')
```

Routine which is called by the above program:

```
% function for calculating the analytical value of dTmax. Parameters are
% the Péclet number Pe, the heat transfer number A, the porosity phi,
% the inital temperature difference between bottom and top dT, the domain
% size H (distance at which top boundary conditions are applied) and the
% distance from bottom z.

function [dTmax]=dTmaxcalc(Pe,A,phi,dT,H,z)

r1 = (Pe-sqrt(Pe^2+4*A/(1-phi)))/2;
r2 = (Pe+sqrt(Pe^2+4*A/(1-phi)))/2;
dTmaxhighA = (1-phi)*Pe/A*dT/H;
alpha = dTmaxhighA/(r1/r2*exp((r1-r2)*H) - 1);
beta = dTmaxhighA/(r2/r1*exp((r2-r1)*H) - 1);
dTmax = alpha*exp(r1*z) + beta*exp(r2*z) + dTmaxhighA;
```