T1dH: A 1-d code for the calculation of Heat conduction with Earth Science Applications

Evangelos Moulas[1*] & Simon Schorn[1]

[1]Institute of Geosciences & Mainz Institute of Multiscale Modeling (M[3]ODEL), Johannes-Gutenberg University Mainz, Mainz, Germany

*Corresponding author (evmoulas@uni-mainz.de)

This documentation describes the main features of T1dH (**T**emperature diffusion in **1d** using the enthalpy (**H**) method). T1dh is set of MATLAB routines that can be used to calculate the temperature evolution in 1 dimension using the enthalpy method. This approach allows for the incorporation of latent-heat effects on cooling/exhuming regions. The routines must all be placed in a common folder. These routines are:

- T1dh.m
- Make_Enthalpy.m

and the main code can be run by typing T1dh in the MATLAB command window. The Make_Enthalpy.m function provided was developed for a particular application. Other applications may require a modification of this function.

The code has been written in general form using functions that would allow the more transparent presentation of the results. More technical details follow below. The software and the present documentation are provided free of charge[1] and they are mostly intended for research and teaching purposes. At this point, all the provided routines have been tested for compatibility with OCTAVE.

Current Version: 1.0 (23-April-2024) – doi: 10.5281/zenodo.11046566

In case of questions please send an email to evmoulas@uni-mainz.de

[1] Creative Commons Attribution 4.0 International

Contents

Introduction

The purpose of the present document is not to provide a detailed review of the literature on thermal diffusion or the enthalpy method but to provide a concise introduction on the methods that are implemented in T1dH and to show the respective governing equations. We have tried to keep the text to a minimum and show how the program can be easily used. Our algorithms have been evaluated and the results have been compared to various analytical solutions (benchmarks). The benchmark comparisons and a detailed description of the code inputs are given in the following parts of this documentation.

Governing Equations and solution procedure

The partial differential equation that describes the evolution of enthalpy in 1 dimension is (e.g. Powers, 2016):

$$\rho \frac{dH}{dt} = \frac{\partial}{\partial z}\left(k \frac{\partial T}{\partial z}\right) + \frac{dP}{dt} + A \tag{1}$$

where t is time, z is the spatial direction (here taken as depth; increasing downwards), $H(z,t)$ indicates the mass-specific enthalpy (in J/kg), and $k{:}\, k(z,t)$ represents the heat conduction coefficient. All variables are given in SI units unless otherwise specified. In addition, $\rho{:}\,(z,t)$ is the density of the material, $P{:}\, P(z,t)$ is the negative mean stress (pressure) and $A{:}\, A(z,t)$ is the volumetric rate of radiogenic heat production. The terms ρ, k and A are assumed to be known functions of depth (z) and in our case ρ, k are considered constant. The term dP/dt denotes the adiabatic heating term. This term takes negative values in case of cooling.

As a first approximation, we can consider that the pressure is lithostatic, i.e. $P \approx \rho g z$ (g being the gravity acceleration). Then, the pressure time derivative is given by the following formula (for constant density, velocity and gravity acceleration):

$$\frac{dP}{dt} \approx \rho g \frac{dz}{dt} = \rho g v_z \tag{2}$$

At this point we should mention that the total derivative operator d/dz includes the advection contribution, that is:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + v_z \frac{\partial}{\partial z} \tag{3}$$

For Eulerian coordinate systems, and for the case where temperature (and enthalpy as well) is fixed at the surface of the Earth (Dirichlet boundary condition), eq. [3] can be written as:

$$\frac{d}{dt} = \frac{\partial}{\partial t} - v_{exh}\frac{\partial}{\partial z} \tag{4}$$

where v_{exh} is the exhumation rate (with units of velocity). Using eqs. [2,4], equation [1] becomes:

$$\frac{dH}{dt} = \frac{1}{\rho}\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) - g v_{exh} + \frac{A}{\rho} \tag{5}$$

Equation [5] is the main equation that is solved in T1dH.m. We use an explicit time integration scheme (forward Euler scheme) and use eq. [5] to update for the enthalpy (H). Then, the temperature is updated by interpolation using a known H: $H(T)$ relation.

The explicit form for the enthalpy function $H(T)$ depends on the particular application. In this work, we consider that the specific enthalpy has two major contributions, one from the heat capacity term and one from the latent heat of mineral reactions/melting. The heat capacity contribution to the enthalpy is given by:

$$H_0 = C_p T \tag{6}$$

In addition, the latent heat contribution is given by a logistic function of the form:

$$\Delta H = \frac{LH}{1 + exp\left(-\lambda_{re}(T - T_{re})\right)} \tag{7}$$

where T_{re} is the temperature of the reaction/melting and λ_{re} is a parameter defining the width of the reaction/melting zone. For the case of melting this zone is also referred to as the "mushy" zone. LH is the latent heat of reaction/melting in units of J/kg. T_{re} and λ_{re} are given by the following formulas:

$$T_{re} = 0.5(T_l + T_s) \tag{8a}$$

$$\lambda_{re} = \frac{10}{0.5(T_l - T_s)} \tag{8b}$$

In our specific case where we investigate the latent heat during magma crystallization, T_l and T_s represent the liquidus and solidus temperatures. For other applications (e.g. latent heat of hydration reactions) T_{re} and λ_{re} have to be specified by the user depending on the particular thermodynamic relations (e.g. Schorn et al., 2024).

The final expression for the enthalpy is given by:

$$H = H_0 + \Delta H \tag{9}$$

The boundary conditions that have been implemented for the bottom are Dirichlet or Neumann (flux-specific) and can change accordingly. The top boundary condition assumes that the temperature at the surface is constant and equal to 0 °C (Dirichlet boundary condition).

Benchmarking

We have performed various benchmarks to verify the accuracy of our results. For all the benchmark solutions we have assumed that $k = 3$, $C_P = 1050$ and $\rho = 2700$. Initially, we consider the classic half-space cooling problem that has a temperature evolution given by the following form:

$$T = T_b + \Delta T erf\left(\frac{z}{2\sqrt{Dt}}\right) \tag{10}$$

where T_b is the boundary temperature value (1450°C) and ΔT is a temperature difference (here (-)100°C) from that value. The thermal diffusivity is given by $D = k/(\rho C_P)$. The results are shown in the figure below:

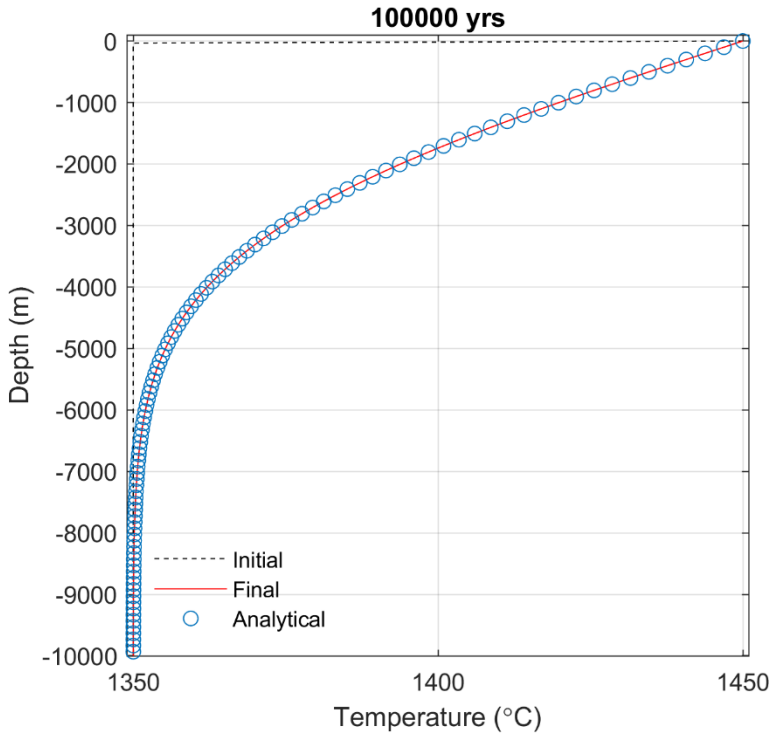

Fig. 1 Comparison of the numerical and the analytical solution for the half-space cooling problem. The solution is evaluated for the time of 100,000 yrs.

Note that the previous solution is not accurate for large timescales because our domain is finite and the analytical solution assumes boundary conditions at infinity. For this case, the contributions from latent heat, radiogenic heat, adiabatic heat, and exhumation are ignored.

An additional benchmark solution that can be used is the analytical solution for the steady state exhumation assuming Dirichlet boundary conditions. The solution can be found in Braun et al., (2006, p. 79) and it's given by the equation below:

$$T(z') = \frac{1 - exp(-Pe \cdot z')}{1 - exp(-Pe)} \Delta T \tag{11}$$

where Pe is the Peclet number given by $Pe = \frac{v_{exh} L}{D}$, L is the total domain length, $z' = \frac{z}{L}$, and ΔT is the temperature difference between the lower and upper boundary (here 500 °C). Note that since eq. [11] is valid for the steady state, we calculated the numerical solution for a sufficiently large time duration. The result is shown in the following figure:

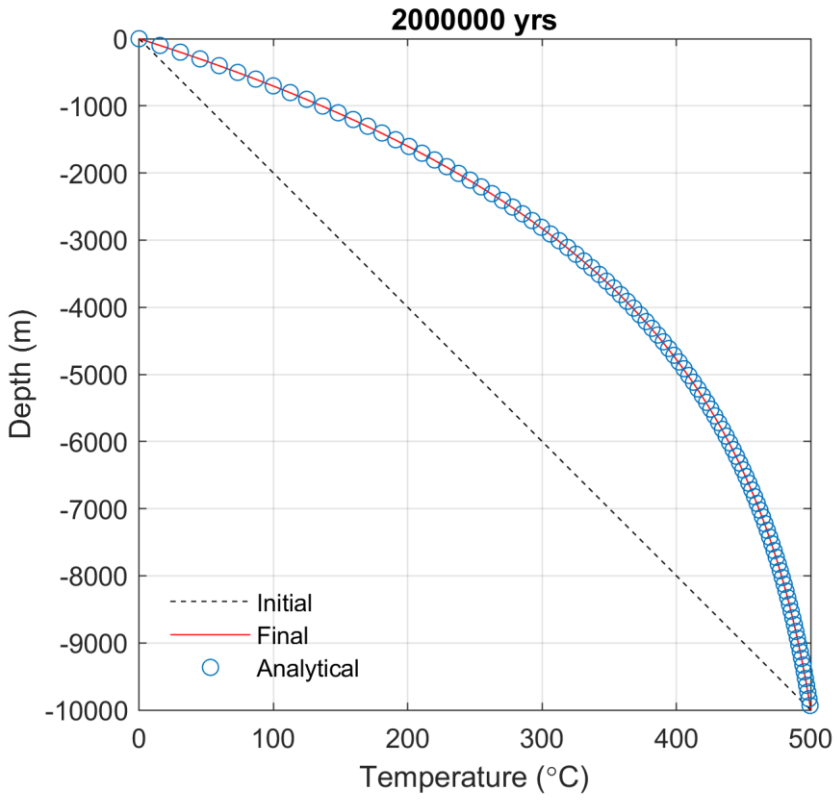

Fig. 2 Comparison of the numerical and the analytical solution for the steady-state exhumation problem. The solution is evaluated for 2 million years.

Note that for the previous example (Fig. 2), the adiabatic and radiogenic heat contributions were neglected.

An additional benchmark that we can calculate is the classic Stefan problem (Turcotte & Schubert, 2014, p. 196). In this case, the initial condition is similar to the half-space cooling problem (Fig. 1). However, in our case, we consider a latent heat of fusion given by eqs. [7-9]. The classic Stefan problem considers the discontinuous enthalpy at the melting temperature, for this reason, and only for the purpose of the benchmarking, we have considered that $T_l = 1350.05$ °C and $T_s = 1350.00$ °C and LH =400,000 J/kg. The results are shown in Fig. 3 below.

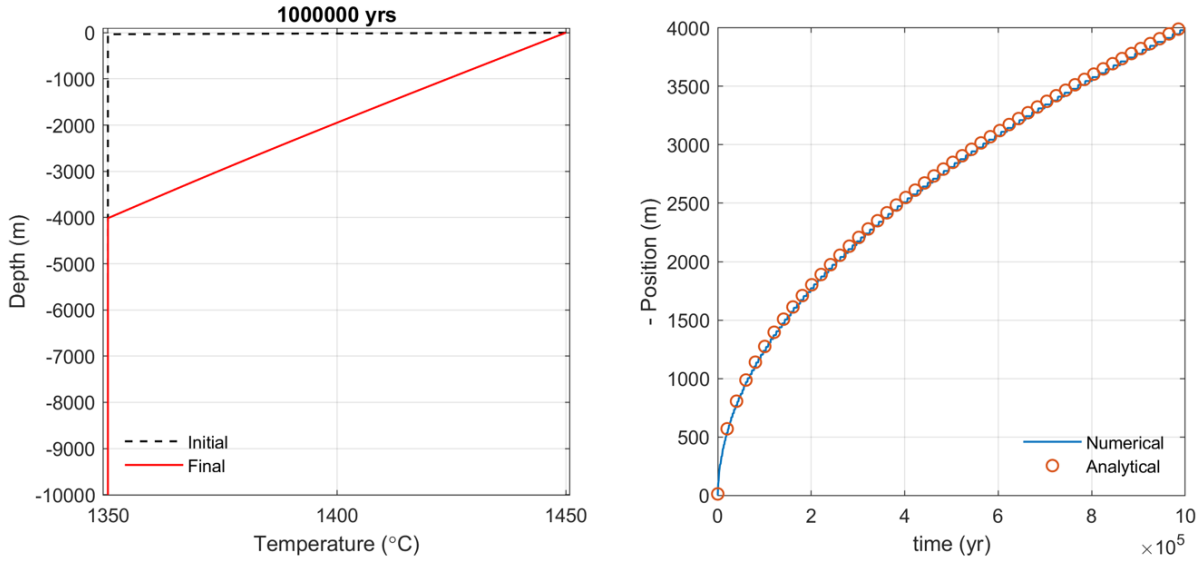

Fig. 3 Left panel: Temperature distribution after 1 million years. Right panel: The position (in depth) of the solid/liquid interface as a function of time. For the calculation of this benchmark both the spatial resolution and the resolution needed to describe eq. [7] where increased from the default values.

The analytical solution for the migration of the solid/liquid front is given by:

$$X(t) = 2\lambda\sqrt{Dt} \tag{12}$$

where λ is given by the Stefan condition:

$$\frac{C_P}{\sqrt{\pi}\cdot LH}(T_m - T(0,0)) = \lambda \cdot exp(\lambda^2)\cdot erf(\lambda) \tag{13}$$

Equation [13] is a transcendental equation and needs to be solved iteratively.

Code description

The main routine of T1dH is T1dH.m. To understand the code, we provide short descriptions of specific code snippets (shown below) that can help the user understand the inputs needed. In the beginning of the code the user needs to specify if the plots need to be shown in every iteration (plot_on = 1), and if the thermal history needs to be saved (saveTt = 1)[2]. In addition, the basal boundary condition can be chosen using Bas_f = 1 (for Neumann boundary condition) or Bas_f = 0 (for Dirichlet boundary condition). In any case, the values of the temperature or its gradient at the boundary conditions are as in the initial condition.

```
%Flags
plot_on = 1;                    %Activate Plot while calculating
plot_f  = 1;                    %plot final
saveTt  = 0;                    %Save Marker T-t
%BC handles
Bas_f   = 0;                    %Basal flux [0 or 1]: off/on
%Numerical Handles
nout    = 25;                   %plot every nout steps (if plot_on==1)
```

Physical parameters

In the next part of the code the user can specify the material parameters, the exhumation rate and the total run duration (k, A, ρ, C_P, v_{exh}). Note that for transparency, v_{exh} is provided in mm per year (VE_mm).

```
%=========================================================================
sec2yr = 60*60*24*365;          %Shortcut - how many secs in yr
g      = 9.81;                  %Gravity
A      = 1e-6;                  %Volumetric Heat Production (micrW/m3)
k      = 3;                     %Conductivity
Tsurf  = 0;                     %T surface
Qs     = 150*1e-3;              %Surface Heat Flux
rho    = 2700;                  %Density
Cp     = 1050;                  %Reference Heat Capacity
VE_mm  = 0;                     %Exhumation (must be>=0)
```

In case of two layers with different properties (e.g. different values of A) are needed, the user can specify the depth of the top layer where the material properties and/or Temperature change (zup). Note that, for our specific application, we have used zup to specify the full length of the domain (zmax variable; described further below).

[2] By setting these variables to zero the results will not be saved.

In the case where the initial temperature and radiogenic heat distribution are distinctly different in the top layer, they must be specified in the lines shown below:

```
zup     = 1000;                    %Thickness (from top) of upper layer
Tup     = 1350;                    %T of top layer (NaN if not used)
Aup     = 1e-6;                    %Radiogenic (layer)
```

In our case, the initial domain is Ndom times the dept of zup. Therefore, it is advisable to keep using zup as a variable even if the lines that define Tup and Aup are commented (see code snippet above). Finally, if the thermal history of a particular point is needed (e.g. if saveTt = 1), the initial coordinates of this point can be specified (zmark).

The relation of enthalpy and temperature is created in the following lines using the function make_Enthalpy.m.

```
%Preprocessing thermodynamic properties --------------------------------
Tdb     = linspace(0,1800,50000);          %T range
Hdb     = make_Enthalpy(Cp,Tdb,LH,Tl,Ts);  %Make Enthalpy array
Cpeff   = diff(Hdb)./diff(Tdb);            %Calculate effective Cp
```

Numerical parameters[3]

The numerical (spatial) resolution can be increased by increasing the variable nx shown below (nx must take integer values). Note however that very large values of nx will lead to slower computational performance due to the timestep restrictions required by the explicit time stepping (CFL condition; Courant et al., 1928).

```
%Numerics ------------------------------------------------------------------
nx      = 100;                          %resolution
zmax    = Ndom*zup;                     %Maximum depth
dz      = zmax/(nx-1);                  %discretization
z       = 0:dz:zmax;                    %domain
```

Setting the initial temperature profile (initial condition)

The initial temperature condition is specified using the formula for the steady state temperature distribution. This formula reads:

$$T(z) = T_{surf} + \frac{Q_s}{k} z - \frac{A}{2k} z^2 \qquad (14)$$

where Q_s is the surface heat flux that is specified in the beginning (Qs).

[3] Note that the maximum depth of the domain is not a numerical parameter, but it is calculated from Ndom that was defined previously.

Running the code

Once all the previous parameters are specified the user can type T1dh in the Matlab (or Octave) command line. In case where the saveTt option is set to 1, then the results of the temperature evolution of the marker will be stored as a mat file. These results include the following variables:

Tpath: the temperature values of the marker's history (in K)

Ppath: the pressure values of the marker's history (in Pa)

tpath: the time values of the marker's history (in years)

CRate: the effective cooling rates provided in K/year (same as °C/year)

tpathc: the time values corresponding to the cooling rate array[4] (in years)

Using T1dH in OCTAVE

Although T1dH was written originally in MATLAB, compatibility with octave has been checked and the codes work normally. The only issues are related to labeling during plotting. In that case, the option "'intepreter','latex'" and he symbol "$" must be deleted from the axis labels/titles.

Acknowledgments

E.M. acknowledges the German Research Foundation (DFG; project number 512790090) for financial support in relation to the development of the methodology and the numerical methods. S.S. acknowledges the Alexander von Humboldt foundation for financial support.

[4] The cooling rates were calculated using numerical differentiation. Therefore, the array CRate has a different size than the array Tpath. For this reason, tpath should be used (e.g. in plotting) with Tpath and Ppath whereas tpathc should be used with CRate.

References

Braun, J., Beek, P. van der, & Batt, G. (2006). *Quantitative Thermochronology: Numerical Methods for the Interpretation of Thermochronological Data*. Cambridge University Press; Cambridge Core. https://doi.org/10.1017/CBO9780511616433

Courant, R., Friedrichs, K., & Lewy, H. (1928). Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, *100*(1), 32–74. https://doi.org/10.1007/BF01448839

Powers, J. M. (2016). *Combustion Thermodynamics and Dynamics*. Cambridge University Press.

Schorn, S., Moulas, E., & Stüwe, K. (2024). Exothermic reactions and 39Ar−40Ar thermochronology: Hydration leads to younger apparent ages. *Geology*. https://doi.org/10.1130/G51776.1

Turcotte, D. L., & Schubert, G. (2014). *Geodynamics* (3rd ed.). Cambridge University Press.