Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-341-AC1, 2019

Interactive comment on "Beo v1.0: Numerical model of heat flow and low-temperature thermochronology in hydrothermal systems" *by* Elco Luijendijk

Elco Luijendijk

elco.luijendijk@geo.uni-goettingen.de

Received and published: 31 July 2019

article

Reply to review by Manolis Veveakis

Authors' note: The reviewer comments are reproduced here. The replies can be found below each comment and are *italicized*.

My apologies for the late review and the wrong post as a comment. I have read the manuscript and since I am not an expert in the field of thermochronology, I will comment on the mathematical and numerical formulation. The paper presents a code solving an advection-diffusion equation for temperature, together with an advection-diffusion equation for helium concentration which is treated as a tracer and complex boundary conditions. The manuscript is overall well written and should be published after addressing minor issues. My main comments are on section 2.

In this section, the author is presenting a complicated combination of constitutive laws and empirical relationships to provide a better boundary condition for the temperature, accounting for the air layer with its humidity and etc. Despite putting all this detail in the BC however, the author is ignoring phase transformations like vaporisation, which will transform his temperature equation from quasi-linear advection-diffusion to nonlinear advection-diffusion-reaction equation. Rather than that, he is using the boiling temperature curve to cap the temperature. Although he discusses his choice on page 6, line 1-5, the importance of implementing a complicated BC instead of accounting for this mechanism is not obvious to me. I would appreciate if the author could comment on that in the revised version.

Reply: One of the motivations of writing this model code was to be able to use a heat flow model to quantify what type of hydrothermal system is needed to explain measured discharge temperatures in thermal springs. However, the modelled temperature of a thermal spring is very sensitive to the land surface boundary condition, as discussed in section 2.1 and shown for instance in Figure 6b. Therefore the decision was made to implement a relatively complex but realistic boundary condition that follows standard formulation of land surface heat flux in the meteorology literature. Note that while the boundary condition is relatively complex in the sense that it consists of a large number of equations, in practice the boundary condition places only moderate computational demands. In the code the boundary condition involves calculating a heat transfer coefficient as a function of the land surface temperature once every modelled timestep for the land surface nodes only. This is a relatively straightforward series of equations *that was easily vectorized with numpy. The revised manuscript includes a motivation for using this boundary condition in section 2.2:*

"The motivation for including the series of equations below in the model code is to simulate realistic land surface and spring temperatures in transient models. Note that the implementation does not place high computational demands, the equations are evaluated for land surface nodes only and in contrast to the heat flow equations in section 2.1 these do not require numerical methods."

In contrast, including phase transitions would necessitate solving an additional equation to calculate phase transitions for each node in the model domain. If I am correct this would also require iterative coupling with the advection-diffusion equation. I have not come up with a computationally efficient way to include this in the model code, and suspect that attempt at this will slow down the model code significantly and may also place additional constraints on grid resolution. Therefore I opted to not invest time in adding multi-phase flow to the model code for now. This point has been clarified in section 2.3 by these additional lines:

"Note that the choice for capping modelled temperature by the boiling temperature avoids the high computational expense of including phase transformations in the model code, which would necessitate solving an additional equation for each node and each timestep, and which would likely place more constraints on spatial discretization and timestep size."

In addition, I haven't understood if the code can handle strongly advecting cases, and if yes what kind of unwinding has it been used? All the examples presented seem to be strongly diffusive.

Reply: The code itself relies on a published finite element code escript for solving the advection-diffusion equation. The implementation that is used is sensitive to numerical *instability that is introduced by advection. To my knowledge escript does not include any unwinding for solving the advection equation, and I did not implement any additional unwinding in the numerical procedure myself. In practice the numerical stability was found to be controlled by timestep size. The solution that was adopted was to start with test models in which the timestep is limited by the CFL condition and subsequently increase the timestep to find the maximum stable timestep size.*

Other than that, I would appreciate if you could explain in more details the transition from Eq. 3 to Eq. 4. I got a bit lost with the nomenclature.

Reply: I agree that this was not so clear in the previous version of the manuscript. The revised version of the manuscript contains a more extensive discussion and a description of which variables in eq. 3 and 4 were considered equivalent to which variables in Fouriers law.