Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-140-AC3, 2020

Geoscientific
Model Development
Discussions

Interactive comment on “*HydrothermalFoam* v1.0: a 3-D hydro-thermo-transport model for natural submarine hydrothermal systems” *by* Zhikui Guo et al.

Zhikui Guo et al.

zguo@geomar.de

Received and published: 21 September 2020

We thank the reviewer for his time, precise summary, and positive evaluation of the manuscript.

Reply to reviewer’s comments, all reviewer comments are in blue and our replies are in black.

1. l72-74: It is important to mention other initiatives that use OpenFOAM to solve flow and transport in porous media with Darcy-like solvers, e.g. Horgue et al. 2015 and Orgogozo et al. 2015

   Yes, thanks for pointing that out - we have added the corresponding references to the manuscript.

2. l84: I think it is important here to mention that the PDEs are solved implicitly but sequentially.

   We have rephrased the sentence to mention the implicit and sequential scheme. The line 84 is rephrased as " *All the partial differential equations are solved implicitly in the framework of OpenFOAM and in a sequential scheme, and the thermal-physical models are developed using a pure water EOS.*"

3. l96: strictly speaking, in fluid mechanics, in laminar flow regime, there is inertia. Darcy’s law corresponds to the creeping flow regime when inertia is negligible compared with viscous forces.

   Good point. We have clarified that and now refer to creeping flow in the main text.

4. Eq (2) : the right-hand side (=0) is missing

   We have fixed the issue in Eq (2).

5. l104: The compressibility of the rock is neglected as soon as the porosity is removed from the time derivative in Eq (1). Then, it is probably better to mention this hypothesis just after Eq (1).

   Yes, that’s, of course, correct. The model is for constant-in-time porosity, i.e. incompressible grains. We have clarified that right after Eqn. (1).

6. the heat conductivity is noted lamba_r in Eqs (4)-(7) but k_r in the code.

   We have modified the λ_r in Eqs (4)-(7) and table 1 to k_r which keeps consistent with the code.

7. I am not sure that the energy equation in Eq (4) (and then Eq (7)) is exact. It seems correct only if the continuity equation is a divergence-free equation which

is not the case in the paper (this is what appears if you derive the equation from the conservation of the internal energy ddt(e) + div(e U) = div(q) + S with e(p,T)). Moreover, it is not clear whether the two last terms of the right-hand side are necessary as they seem orders of magnitude lower than the other terms. Can you comment on that? Give an estimation of the weight of these terms and explain in which cases it is important to consider them?

As both reviewers had concerns about the assumptions and validity range of the energy equation, we have posted the full derivation as a separate author comment. We hope that that derivation is sufficiently clear to resolve questions about which terms are inside and outside the spatial and temporal derivatives.

That derivation is starting from the equation of change of internal energy, just as the reviewer pointed out. We have kept the pV and dissipation terms in the starting equation, both for completeness and because they actually matter for some submarine settings. But please see the full derivation for details.

Concerning the pV and dissipation terms (the last two terms in the energy equation): We did not find very many published studies but there is a nice pioneering paper by Garg and Pritchett ("On pressure-work, viscous dissipation and the energy balance relation for geothermal reservoirs", Advances in Water Resources, 1977). They show for geothermal systems that one either should keep both terms or none of them as they point to opposing directions. They also point out that for highly compressible cases (pure vapor, or supercritical fluid close to the critical point) they actually matter. We have made our own tests and find that also for large (say >5km) vertical extents (e.g. fault controlled systems at slow spreading ridges), these terms also matter.

That said, it is important to point out that the $-\left(\frac{\partial ln\rho_f}{\partial lnT}\right)_p\left(\varepsilon\frac{\partial p}{\partial t}+\vec{U}\cdot\nabla p\right)$ term is not exactly equal to pressure volume work but also contains the pressure dependence of enthalpy (see derivation). That terms again matters, for example, when

simulating fluid flow close to the critical point. Benchmarks C,D, where a fluid of constant temperature is flowing along a pressure gradient, show a drop in temperature on the lhs of the domain, which is partly related to this effect.
We have added a reference to the Garg and Pritchett paper in the main text.

8. Assuming that the equation is correct, the numerical treatment presented in Listing 10 can be improved by making the last term implicit with fvm::Sp(alphaP*(...) , T), which should lead to better numerical stability.

   Yes we agree. We have changed the last term of temperature equation to `fvm::Sp(alphaP*(porosity*fvc::ddt(p)+(U&fvc::grad(p))),T)` in the source code and Listing 1 in the manuscript.

9. l139: “adapt” instead of “adopt”

   Corrected.

10. l159-160: the sentence "since it is not straightforward to impose fluid velocities on boundaries" is not clear. Actually, here you just transport a velocity value into a boundary condition on the pressure gradient because your solver solves a pressure equation only.

    The sentence has been revised to "A similar boundary condition called `hydrothermalMassFluxPressure` is defined for the pressure field to prescribe a mass flux into the modeling domain."

11. l166: what is the difference between submarinePressure and OpenFOAM’s PrghPressure?

    `submarinePressure` is a kind of fixed value boundary condition for pressure `p` , and it is specially designed for seafloor boundary of submarine hydrothermal problem. It provides hydrostatic pressure according to bathymetry or coordinate of seafloor patch. If user want use `submarinePressure` to set seafloor pressure boundary condition, the `y` coordinate must be negative because assuming `y` of sea level is zero. However, `prghPressure` is designed for `p_rgh` . The `submarinePressure` is much more straightforward and easier to use for submarine hydrothermal modeling. The detail difference between them are shown below,

- *prghPressure*

  This boundary condition provides static pressure condition for `p_rgh`, calculated as:

  $$p_{rgh} = p - \rho g(h - hRef) \tag{1}$$

  where

  - p_{rgh}: Pseudo hydrostatic pressure [Pa]
  - p: Static pressure [Pa]
  - h: Height in the opposite direction to gravity
  - $hRef$: Reference height in the opposite direction to gravity
  - ρ: density
  - g: acceleration due to gravity [m/s2]

  Code snippet

  ```
  this->operator==
  (
      *this - rhop*((g.value() & this->patch().Cf())
      - ghRef.value())
  );
  ```

- *submarinePressure*

  This boundary condition provides fixed hydrostatic pressure condition for `p` at **seafloor** boundary patch, which is derived from *fixedValueFvPatchScalarField* and calculated as:

  $$p = p_a + \rho_{sw} * (\vec{g} \cdot \vec{C}_{patch}) \tag{2}$$

  where

  - p: pressure [Pa]
  - p_a: pressure at sea level (or y=0 level), default is atmospheric pressure 10^5 [Pa].
  - ρ_{sw}: seawater density, default is 1013 [kg/m3].
  - $\vec{g}$: gravitational acceleration vector [m/s2]
  - $\vec{C}_{patch}$: face centres (coordinate) of seafloor patch [m]. **Note** that y coordinate of seafloor surface must be negative when using `submarinePressure` boundary condition.

  Code snippet

  ```
  operator==
  (
      p_Atmospheric + rhoValue_ * (g.value() & patch().Cf())
  );
  ```

  Example usage

  ```
  seafloor
  {
      type    submarinePressure;
      rhoValue 1013;  //Optional
  }
  ```

12. l177: more details will be appreciated on the implementation of the thermo-physical model.

    The thermo-physical model is implemented following the general guidelines for a thermo-physical model in OpenFOAM, e.g., `heHydroThermo` is modified from `$FOAM_SRC/thermophysicalModels/basic/heThermo`. Then just call thermal dynamic property calculation function of freesteam library, e.g. `freesteam_rho`, in the implementation function of the thermo-physical model, e.g. `species/water/equationOfState/IAPWSEOSI.H`. More details can be found in the source code.

    We would like to leave it at that level of detail as a more in-depth discussion would distract from the main points of the paper.

13. l194-1999: the sentence about constrainPressure and fixedFluxPressure is very confusing. Why do you need fixedFluxPressure in your simulations? What is the link with the boundary conditions introduced in Section 2.4?

    Actually the `fixedFluxPressure` boundary condition and the `hydrothermalMassFluxPressure` (introduced in section 2.4) are similar, the first one is the OpenFOAM's internal BC which requires `U`, while the second one doesn't need `U`. Although `fixedFluxPressure` boundary condition is not used frequently in our simulation, in order to make the solver compatible with this boundary condition, the function of `constrainPressure` has to be added to update pressure boundary condition before solving `p` in the pEqn.C file. In addition, `constrainPressure` only works when pressure boundary condition is `fixedFluxPressure` , otherwise it doesn't do anything.

14. It is difficult for the potential user to know where is the code. It is scattered on too many platforms. The document mentions at least 3 different locations (Zenodo.org, DockerHub, GitLab). In particular, Zenodo.org and GitLab seem to provide the same code.

    Thanks for these comment, indeed there are two many locations, we have add necessary explanations to each repositories in the code availability section.

    - GitLab repository always contains the latest development version of the source code.
    - Zenodo repository is selected only for the GMD publication because it provides a DOI number. It will be updated in the future if we have new release version of HydrothermalFOAM.
    - Docker Hub repository contains all the runtime environment, e.g. OpenFOAM, Gmsh, python, HYDROTHERM, etc., required by HydrothermalFOAM and benchmark examples in the manuscript. This repository will not be updated in the future unless some new runtime environment is required by HydrothermalFOAM.

    Therefore, the potential user should only follow the GitLab repository for the source code. The Docker Hub version is for the new users who are not familiar with OpenFOAM or doesn't have OpenFOAM.

    We also post an author comment of How to update HydrothermalFoam to describe how fetch and use the latest source code.

15. l354-355, l364-365, l374-375: "can be be", "bechmarks"

    We have corrected all these typo issues in the manuscript.

16. l396 Zenodo.org

    We have corrected this typo in the manuscript.

17. Section 5: I think it will be interested to have a comparison of the simulation time of HYDROTHERM and HydrothermalFoam. Such a comparison can highlight better the importance of using modern computational platforms.

We have compared the simulation time of HYDROTHERM and Hydrothermal-FOAM for 3D benchmark examples described in the manuscript (section 3.2). HydrothermalFOAM is $\sim$24 times faster than HYDROTHERM in serial computing case. In addition, HydrothermalFOAM has parallel computing ability but HYDROTHERM seems not. Therefore, maybe it is not necessary to compare the simulation time of them. While the simulation time of these two solvers for the 3D benchmark examples are listed in table 1 to answer your question.

We'd prefer to not enter that discussion in the main text because it is not an entirely fair comparison and because better performance with respect to HYDROTHERM was not our motivation to develop the Openfoam-based model.

Table 1. Simulation time comparison of HYDROTHERM and HydrothermalFOAM for 3D benchmark models

Simulator	3D Homogeneous model		3D Heterogeneous model	
	Serial	Parallel (4 cores)	Serial	Parallel (4 cores)
HydrothermalFOAM (h)	1.2575	0.4872	0.8081	0.5386
HYDROTHERM (h)	25.0403	-	30.2556	-