Richards Examples

Andy Wilkins
CSIRO

April 29, 2014

Contents

1 Introduction

The Richards' equation describes slow fluid flow through a porous medium. This document outlines input-file examples for the Richards MOOSE code, drawing mainly upon the test suite. There are two other accompanying documents: (1) The theoretical and numerical foundations of the code, which also describes the notation used throughout this document; (2) The test suite, which describes the benchmark tests used to validate the code.

2 The examples

Each example is located in the *test* directory, which has path

`<install_dir>/moose/modules/richards/tests`

or the *user* directory, which has path

`<install_dir>/moose/modules/richards/doc/user`

2.1 Convergence and convergence criteria

As a general rule, the formulation of multiphase flow implemented in MOOSE is quite suitable to solve many different problems. However, there are some situations where the implementation is not optimal, such as the tracking of fronts. MOOSE may converge in these instances, but may take unacceptably short time steps, or it may not even converge at all. In these cases it is probably best to either modify the problem to something more suitable, or use another code.

For example, in simulations with sharp fronts, the user should ask whether it is truly necessary to accurately track the sharp front, or whether similar results could be taken by smoothing the fronts by using slightly different initial conditions, and/or by modifying the effective-saturation relationships, by making the van Genuchten parameter α smaller, for instance. If tracking sharp fronts is vital to the problem then a code specifically designed to solve such problems will do it better than MOOSE, or perhaps MOOSE could be used with an ALE front-tracking algorithm.

Below I list some general pointers that may help with problems that MOOSE is struggling with.

- Effective saturation curves that are too "flat" are not good. For example, the van Genuchten parameter $m = 0.6$ almost always gives better convergence than $m = 0.9$.
- Effective saturation curves that are too "low" are not good. For example, the van Genuchten parameter $\alpha = 10^{-6}\,\mathrm{Pa}^{-1}$ almost always gives better convergence than $\alpha = 10^{-3}\,\mathrm{Pa}^{-1}$. Compare, for instance, the tests `buckley_leverett/bl21.i` and `buckley_leverett/bl22.i`.
- Any discontinuities in the effective saturation, or its derivative, are bad. I suggest using van Genuchten parameter $\alpha \geq 0.5$ for problems with both saturated and unsaturated zones.
- Highly nonlinear relative permeability curves make convergence difficult in some cases. For instance, a "power" relative permeability curve with $n = 20$ is much worse numerically than with $n = 2$. See if you can reduce the nonlinearity in your curve.

- Any discontinuities in the relative permeability, or its derivative, are bad. For most curves coded into MOOSE this is not an issue, but I recommend the `RichardsRelPermVG1` curve over `RichardsRelPermVG`, since the former is smooth around $S_{\rm eff} = 1$. See `recharge_discharge/rd01.i` in the tests directory for an example of this. I also suggest using the "power" covers over the "VG" curves.

Choosing reasonable convergence criteria is very important. The Theory Manual contains a section that explains the *minimum* residual that a user can expect to obtain in a model. However, this minimum is usually much smaller than what is important from a practical point of view. If a tiny residual is chosen, MOOSE will spend most of its time changing pressure values by tiny amounts as it attempts to converge to the tiny residual, and this is a waste of compute time. This problem may be amplified if adaptive time-stepping is used since MOOSE doesn't realise that most of the compute time is spent "doing nothing", so keeps the timestep very small. So, here are some guidelines for choosing an appropriate tolerance on the residual.

1. Determine an appopriate tolerance on what you mean by "steadystate". For instance, in a single-phase simulation with reasonably large constant fluid bulk modulus, and gravity acting in the $-z$ direction, the steadystate solution is $P = -\rho_0 g z$ (up to a constant). In the case of water, this reads $P = -10000z$. Instead of this, suppose you would be happy to say the model is at steadystate if $P = -(\rho_0 g + \varepsilon)z$. For instance, for water, $\varepsilon = 1\,\text{Pa.m}^{-1}$ might be suitable in your problem. Then recall that the residual is just

$$R = \left| \int \nabla_i \left(\frac{\kappa_{ij} \kappa_{\rm rel} \rho}{\mu} (\nabla_j P + \rho g_j) \right) \right| \tag{2.1}$$

   Evaluate this for your "steadystate" solution. For instance, in the case of water just quoted, $R = V|\kappa|\rho_0/\mu\varepsilon = V|\kappa| \times 10^6\varepsilon$, where: V is the volume of the finite-element mesh, and I have inserted standard values for ρ_0 and μ.

2. In the previous step, an appropriate tolernace on the residual was given as $V|\kappa|\rho_0\varepsilon/\mu$. However, this is often too large because of the factor of V. The previous step assumed that the solution was incorrect by a factor, ε, which is constant over the entire mesh. More commonly, there is a small region of the mesh where most of the interesting dynamics occurs, and the remainder of the mesh exists just to provide reasonable boundary conditions for this "interesting" region. The residual in the "boring" region can be thought of as virtually zero, while the residual in the "interesting" region is $V_{\rm interesting}|\kappa|\rho_0\varepsilon/\mu$. This is smaller than the residual in the previous step, so provides a tighter tolerance for MOOSE to strive towards.

3. In the previous steps, I've implicitly assumed κ is constant, ρ is virtually constant at ρ_0, only a single-phase situation, etc. In many cases these assumptions are not valid, so the integral of Eqn (2.1) cannot be done as trivially as in the previous steps. In these cases, I simply suggest to build a model with initial conditions like $P = -(\rho_0 g + \varepsilon)z$, and just see what the initial residual is. That will give you an idea of how big a reasonable residual tolerance should be.

2.2 Two-phase, almost saturated

If a two-phase model has regions that are fully saturated with the "1" phase (typically this is water), then the residual for the "2" phase is zero. This means the "2"-phase pressure will not change in those regions, potentially violating $P_1 \leq P_2$. If the "2" phase subsequently infiltrates to these regions, an initially crazy P_2 might affect the results. This sometimes also holds for almost-saturated situations, depending on the exact simulation.

In these cases, one way of avoiding problems is to implement the constraint $P_1 \leq P_2$ using a `RichardsMultiphaseProblem` Problem object. This gets around the problem of chosing the a in the penalty term described in the next paragraph. See Section 2.4 for more comments.

Another way is to add a penalty term to the residual to ensure that $P_1 \leq P_2$. An example can be found in the tests directory `pressure_pulse/pp22.i`. The choice of the a parameter is sometimes difficult: too big and the penalty term dominates the Darcy flow; too small and the penalty term does nothing. In both cases, convergence is poor as the penalty term switches on and off during the Newton-Raphson procedure. The documentation for `RichardsPPenalty` describes how to set a (run MOOSE with a `- -dump` flag).

The penalty term should *not* be used unless absolutely necessary as it will lead to poorer convergence characteristics. In many cases it is not necessary.

2.3 An excavation

In the test directory `excav/ex01.i` and `excav/ex02.i` contains a single excavation, and the associated mass flux and mass balance.

2.4 Bounding porepressure

Occasionally it might be useful to bound porepressure. The test `buckley_leverett/bl22.i` has "bounds" that do this. Note that:

- The convergence is likely to be much slower when using bounds
- The `-snes_type` must be set to `vinewtonssls` (see the `[Preconditioning]` block of the aforementioned test.
- The command line must contain the argument `--use-petsc-dm`.

The "bounds" just described uses PETSc to enforce $P \geq a$, for some fixed a. In two-phase situations, one often wants to enforce $P_1 \geq P_2$. To do this a `RichardsMultiphaseProblem` object can be used. See for example `gravity_head_2/gh_bounded_17.i`, which may be compared to `gravity_head_2/gh_lumped_17.i` to see how easy it is to use this in an input file. Note that:

- The convergence is likely to be much slower when using a `RichardsMultiphaseProblem` object.
- The use of this object will likely avoid crazy behaviour of a gas phase's porepressure when the phase is completely or almost completely absent in a region.

- Care must be taken: in the example `gh_bounded_17.i`, I had to use zero residual saturations, otherwise the simulation keeps trying to reduce `pgas` below `pwater` to conserve gas mass. If you see your nonlinear residual not decreasing, you might be using `RichardsMultiphaseProblem` to enforce a constraint which does not make physical sense (eg, it causes nonconservation of mass).