**In Class Problem**

Sept 24, 2020

Write a computer program to solve flow in a 1D reservoir using the finite difference method. The program should allow the user to define the reservoir/fluid properties, final time, number of grid blocks and timestep size, and the numerical method (explicit, implicit, or Crank-Nicholson). The program should also be flexible enough to allow for different boundary conditions (Dirichlet and/or Neumann on either side) in the input file. Use flow/transmissibility units.

1. Verify your code against the solution you obtained in Example 3.4 (using 4 blocks)
2. Use your code to solve Example 3.4 but using *N* = 10 grid-blocks, t = 1 day, and the explicit method. Make a plot of pressure versus position at three different times (choose 3 times that are interesting; a relatively early time, a middle time, and a time close to steady state).
3. On the same plot as part (b), plot the analytical solution (See Chapter 2) at those same three time steps to make a comparison.
4. Repeat part (b) and (c) but using the implicit method
5. Repeat part (b) and (c) but using the Crank-Nicolson method
6. From the criteria for the stability of the explicit method, find the maximum time-step size for which the explicit method is stable for *N* = 10 grid blocks. Then pick a time-step exactly 0.5 days longer than that maximum and plot the simulation results using the explicit method.

Code advice: Create files that do the following:

* Input file with all reservoir, fluid, and numerical properties (I suggest using structures)
* Main/script/driver file that
  + Reads in inputs
  + Loops through all N grid blocks to compute T, B, and Q. (this could be its own function file)
    - Grids *i*=1 and N will be a little different because of boundary conditions
    - Boundary conditions could be Dirichlet or Neumann
  + Loops through time until the final time is reached
* Postprocessing file that creates a plot of pressure versus distance at various times

Note: this is the first reservoir simulator code we will write. If you make this code clean and flexible you can use it as a starting point for future, more complicated codes.