A walk-through of modifying an existing script for a basic solid file based domain

ParFlow Short Course, Friday, May 30, 2019
Notes by Nick Engdahl

The worst way to try and get a PF model running is to try and create a working TCL from scratch. We always start from a working example, usually from the "test" folder. Here we're going to walk through that process for a domain that uses a solid file. Note that there is a bug in the solid file (probably a file corruption from being moved to too many different computers) that prevents us from using this on multiple processors, but everything below works just fine when running in serial and will work with other solid files on any number of processors.

The files for this example are contained within an archive in the "ParFlow_Short_Course" git repository in the "ParFlow_Short_Course/Exercises/East_Inlet" folder. Navigate over to that folder and the working files needed for this example are in the compressed archive "East_Inputs_2019.tar.gz" so step one is to extract the contents of this file. Now from the command line [*comments in brackets at the end are just annotations*]:

```
> gzip -d East_Inputs_2019.tar.gz   [decompress the archive, .gz will
disappear after this command]
> tar -xvf  East_Inputs_2019.tar    [extracts the archive]
```

Or instead of "`gzip -d`", you can use "`gunzip`" to do the same thing. This archive contains a tcl scrip called "East_Inlet_Coarse.tcl" and this will correctly run the domain but *don't open this script now*!! We want to try to build the equivalent and then you can compare the results. The things we need are the slopes and the pfsol file.

During the workshop, we went through the process of inspecting lots of the test cases (located in $PARFLOW_DIR/test) to see what elements they contain that are needed for the kind of domain we want to set up. This includes terrain following grids, indicator files, heterogeneities, solid files, CLM, multiple geometries, and so on. Reviewing those, we identified that the best test case to start form is the "small_domain.tcl" so step one is to make a copy of this and place it in the folder you extracted the archive in. Maybe rename it "East_Test.tcl" if you like so you know it's your working script.

The plan is to go through this script line by line and make any changes necessary to redefine the geometry and grid for the East Inlet model (see Engdahl and Maxwell, 2015, Journal of Hydrology), albeit slightly coarsened to make it feasible on something less than a large compute cluster since that model was originally run on 2048 cores.

Open up the file, either in vim or any other editor and look over the first 48 lines (which is everything above "# The Names of the GeomInputs"). Step one is to read through the example you've selected and understand what it was designed to do. If you don't understand the purpose (geometries, time cycles, boundaries, etc…) you won't be very efficient or effective in translating the scrip into something you need. This particular test is designed to have a variable size, which we don't want, and we also need to redefine the grid for the east inlet domain. We'll worry about modifying parameters after the geometry is fixed. When I set up the East Inlet domain, I chose to use UTM zone 13 (northern hemisphere, centered on Colorado) coordinates so the lower corner of the domain is: x_0=437300, y_0=4450260, z_0=2910. The total size of the domain needs to be 6.4km (along "x") by 5.36km (along "y"), and 1400m vertically, and we'd like cell sizes of dx=dy=40m and dz=25m, so this gives nx=160, ny=134, and nz=56, for a total of just over 1.2 million cells; not a terribly big grid, though we're going to deactivate a lot of those cells soon.

The ComputationalGrid.Lower.* (where * is a place holder for X Y Z) and ComputationalGrid.* (where * is a place holder for NX NY NZ) are exactly as given above and so are the ComputationalGrid.# (where # is a place holder for DX DY DZ). For your own domains you'll obviously use different ones but always ask yourself if the cell sizes are meaningful. Truthfully, these are too coarse vertically, but this is just an example.

If you make those changes, the 9 "set" commands on lines 33-43 can be deleted but for now just comment them out (start each line with "#"). This is another good habit to get in to when modifying existing scripts: don't delete anything until you know your version works!

Moving on to lines 53-71, we see the definition of the geometries. The "background" isn't really necessary but we'll leave it for now. What we need to do next is replace the solid file with our solid file. This is as simple as changing the file name to: "TopesActiveDomain.pfsol." Since I built this solid file, I know I only defined two patches, which are defined in the order "BOTTOM TOP" but looking at the list of patch names on line 70-71 we can just replace the first two with those names and delete the rest of the list. Again, the only reason we know to do this in this case is because I made the solid file; other solid files may use different numbers of patches and patch orders. So comment out line 71 and change line 70 so the set command is:
pfset Geom.domain.Patches   "BOTTOM TOP"   [*Note: you can call these anything you like, it's the order that's most important*]

Moving on, permeabilities are set to the geometry named "domain" so no change is needed there and since the "background" covers all space we can leave the tensor specifications alone too. In fact, everything is OK up to line 136, where the timing information is set. The only change we're going to make is to set the permeability to 0.01 to make the model is easier to solve, so change line 82 to:
pfset Geom.domain.Perm.Value   0.01

Usually the first thing we're interested in doing is just making sure the model sets up the geometry we want correctly, so its OK to use bogus (i.e. no ground truth whatsoever) values during this phase…just don't forget to update these to reflect your actual simulation domain later on!

This timing info might be OK as it is, but let's reduce this so that we can let the model run for a shorter time, just to make sure things seem reasonable. By that, I mean that right now we only care about verifying that the geometry has been built correctly; we're not going to try to spin up this domain given its complexity. Use the following settings:

```
pfset TimingInfo.BaseUnit          0.1
pfset TimingInfo.StartCount        0
pfset TimingInfo.StartTime         0.0
pfset TimingInfo.StopTime          0.1
pfset TimingInfo.DumpInterval          0.1
pfset TimeStep.Type            Constant
pfset TimeStep.Value           0.1
# pfset TimingInfo.DumpAtEnd           True
```

The DumpInterval could be changed to -1 to ensures every step the solver takes will be output so even if it crashes before reaching out specified output we could still look at the outputs, but here it should be fine.

The next 50 or so lines are just constants, which we'd change later if we planned on running a real problem, and our next change will be to simplify the time cycles. On line 190, we only want the "alltime" cycle so this block will be changed to:

```
pfset Cycle.Names "constant"
pfset Cycle.constant.Names           "alltime"
pfset Cycle.constant.alltime.Length  1
pfset Cycle.constant.Repeat          -1

# pfset Cycle.onoff.Names            "on off"
# pfset Cycle.onoff.on.Length         10
# pfset Cycle.onoff.off.Length        90
# pfset Cycle.onoff.Repeat           -1
```

Before this line 190 had *pfset Cycle.Names "constant onoff"* and lines 195-198 were not commented. This just makes it simpler.

Next we'll modify the boundary conditions. Line 203 uses an internal ParFlow command to fetch all the patch names assigned to the geometry "domain" but really we only care about "TOP" and want everything else to be a zero flux. Any boundary that isn't specified explicitly is assumed to be a zero flux so we're going to cheat like mad. Change line 203 to:
pfset BCPressure.PatchNames   "TOP"   *[Note: if you didn't call this patch TOP you'll need to replace it with the name you chose]*

Then comment out lines 205-228, then one lines 230-323, replace "x-upper" with "TOP" and that's it. Later this can be changed to an overland flux boundary with a specified flux but for now, easy does it.

We can now skip on down to line 267 and think about the change we want to make here. The initial condition is set on the geometry "domain" which is fine, but the patch x-lower doesn't exist. Instead we'll reference our IC from the top using the following:

```
pfset Geom.domain.ICPressure.Value           -30.0
pfset Geom.domain.ICPressure.RefPatch               TOP
pfset Geom.domain.ICPressure.RefGeom                domain
```

And this will put the static water table depth 30m below the top of the model, which is just into the second subsurface layer since our dz=25m.

The boundary condition assigned to the geometry "infiltration" isn't used because infiltration isn't actually a geometry. You can comment it out, or not, but it makes no difference.

Below here are just solver settings, which we'll leave alone for now, so skip on down to line 317 where it says "# Tests." This block is used in test cases to compare a simulation to a known solution from within ParFlow's default directory structure. Since we're outside of that, and since this has no known solution, all of this can be deleted; that's right, this is the one place in this file you can delete something without worrying. Say goodbye to lines 316-350 and we're ready to try this run.

Well, almost, let's change the runname first. Go all the way back up to the top. You can comment out the "size" variable and reset the name to something else, maybe EastInlet. I tend to use underscores in tcl names (i.e. East_Inlet.tcl) and omit them from runnames to make file deletion easier and prevent accidently deleting the main run script. In most runs you'd also change the processor topology but, as I said there seems to be a bug in this solid file that limits this run to 1 1 1, so just stick with that.

One last thing is that we'd like to be able to look at our outputs easily so we're going to add the following somewhere around line 310:
"pfset Solver.WriteSiloPressure   True"
This will create SILO formatted pressure files for each time step/iteration

Now we're ready to go. Run the TCL and open up the output pressure files in Visit as a pseudocolor plot. You'll need to hide the inactive cells to see the real domain so use a "threshold" operator and set the minimum value for pressure to be -999. In the properties menu for the pseudocolor plot, change the scaling option from "use original data" to "use current plot" and rescale the pressures so you can clearly see what's happening in the top layers. It won't match the pressure field you'll see in Engdahl and Maxwell (2015) because the domain isn't spun up, it's homogeneous, and we only ran for a single time step of 0.1 days. If you change the run length to be longer (like 1.0) and look at the last outputs you should be able to convince yourself the pressures are adapting to topography, which is probably easiest to see on the bottom layer of the model. Rotate and rescale the pressures to see that more clearly and you're done.

The art of building solid files can be tricky, but there are some blog posts on domain setup that should help you with that. Also, you can also use solid files with terrain following grids, and use multiple solid files to define different regions, but both of these topics are way beyond the kind of thing we can cover in an introductory short course.