Setting up a watershed model: Little Washita Example part 1

ParFlow Short Course

Workflow Outline

1. Evaluate available model inputs
2. Determine your domain configuration
3. Process topography
4. Setup the subsurface
5. Initialize the model (i.e. spinup)
6. Additional setup for PF-CLM

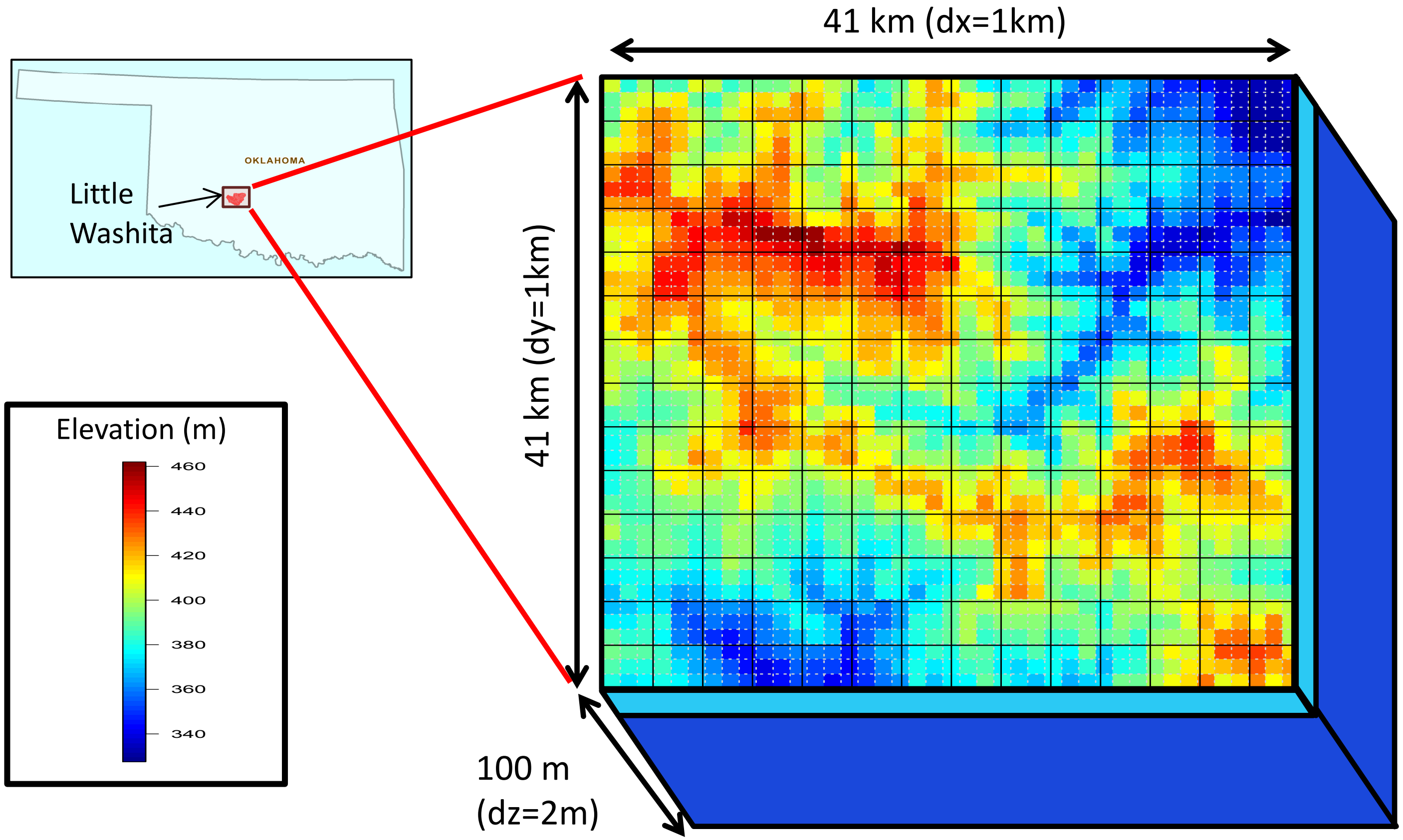
This is also outlined in section 3.1.2 of the manual

2. Determine your domain configuration

1. *What are the questions you want to answer with your model?*

2. *What kind of inputs do you have available to build your model with?*

Lateral resolution and domain extent



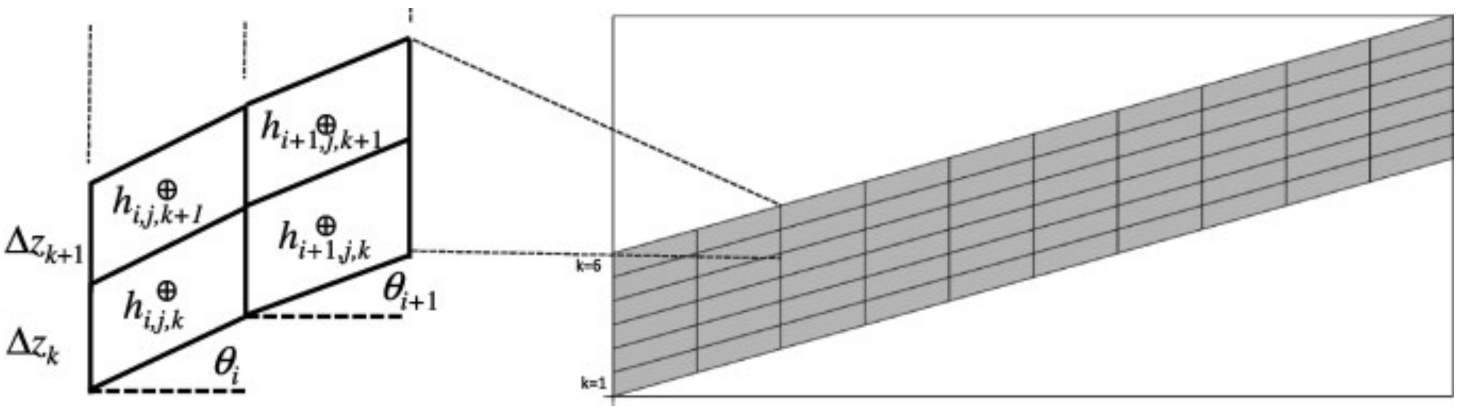
- 1 km lateral resolution
- 2 m vertical resolution extending 100 m

Setting Resolution and domain extent

```
# Define the number of grid blocks in the domain.
model.ComputationalGrid.NX = 64
model.ComputationalGrid.NY = 32
model.ComputationalGrid.NZ = 10

# Define the size of each grid cell. The length units are the same as those on
hydraulic conductivity, here that is meters.
model.ComputationalGrid.DX = 1000.0
model.ComputationalGrid.DY = 1000.0
model.ComputationalGrid.DZ = 200.0

#Locate the origin in the domain.
model.ComputationalGrid.Lower.X = 0.0
model.ComputationalGrid.Lower.Y = 0.0
model.ComputationalGrid.Lower.Z = 0.0

#Declare the geometries that you will use for the problem
model.GeomInput.Names = "solid_input"

#Define the solid_input geometry.
model.GeomInput.solid_input.InputType = "SolidFile"
model.GeomInput.solid_input.GeomNames = "domain"
model.GeomInput.solid_input.FileName = "LW.pfsol"

#First set the name for your `Domain` and setup the patches for this domain
model.Domain.GeomName = "domain"
model.Geom.domain.Patches = "top bottom side"
```

Grid configuration:

– Terrain following grid:
  - Transforms the grid to conform to topography using the slope files
  - Generates a grid with a uniform thickness everywhere
  - Can only be used with Solver Richards and not available with Solver Impes

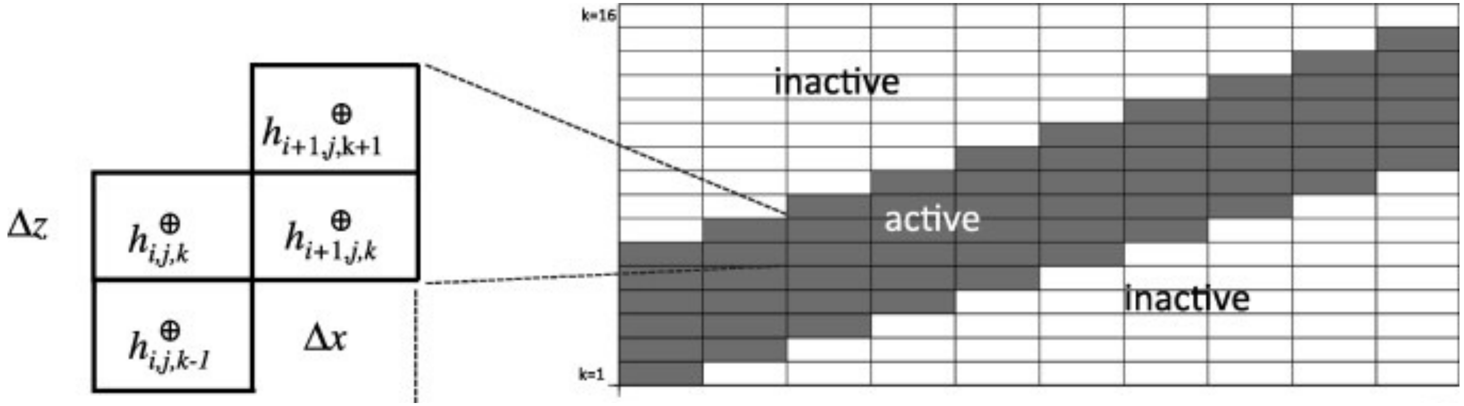


Maxwell, 2013

```
model.Solver.TerrainFollowingGrid = True
model.GeomInput.solid_input.GeomNames = "domain”
model.GeomInput.solid_input.InputType = "SolidFile"
```

Grid configuration:

– Orthogonal grid:
  - Traditional grid
  - Allows for irregular geologic layers and non-uniform watershed depths
  - Requires a pfsol file to define active and inactive cells

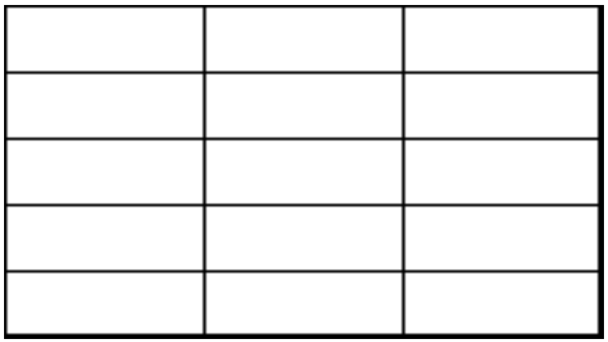


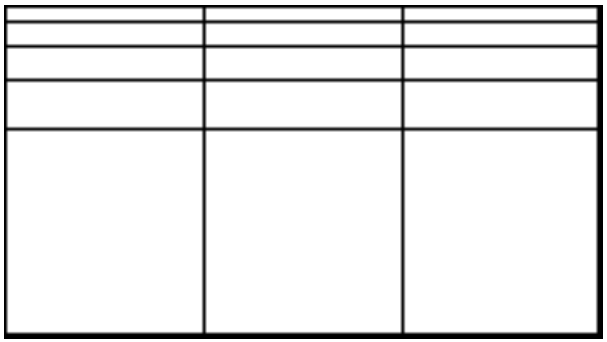

```
model.GeomInput.solid_input.InputType = "SolidFile"
model.GeomInput.solid_input.GeomNames = "domain"
model.GeomInput.solid_input.FileName = "LW.pfsol"
```

http://parflow.blogspot.com/2008/08/patch-order-and-solid-files.html

Grid configuration:

- Constant or variable layer thickness (dz)





- Variable dz allows for thin layers at the surface and thicker layers at the bottom of the domain to maintain high resolution in the upper layers and total domain thickness without too many layers
- See Manual [6.14 dZMultipliers]

Grid configuration:

Variable dz example for a terrain following grid with solid file geometry

```
model.ComputationalGrid.NZ = 10
model.ComputationalGrid.DZ = 200.0

model.Solver.Nonlinear.VariableDz = True
model.dzScale.GeomNames = "domain"
model.dzScale.Type = "nzList"
model.dzScale.nzListNumber = 10

# 10 layers, starts at 0 for the bottom to 9 at the top
model.Cell._0.dzScale.Value = 5
model.Cell._1.dzScale.Value = 0.5
model.Cell._2.dzScale.Value = 0.25
model.Cell._3.dzScale.Value = 0.125
model.Cell._4.dzScale.Value = 0.05
model.Cell._5.dzScale.Value = 0.025
model.Cell._6.dzScale.Value = 0.005
model.Cell._7.dzScale.Value = 0.003
model.Cell._8.dzScale.Value = 0.0015
model.Cell._9.dzScale.Value = 0.0005
```

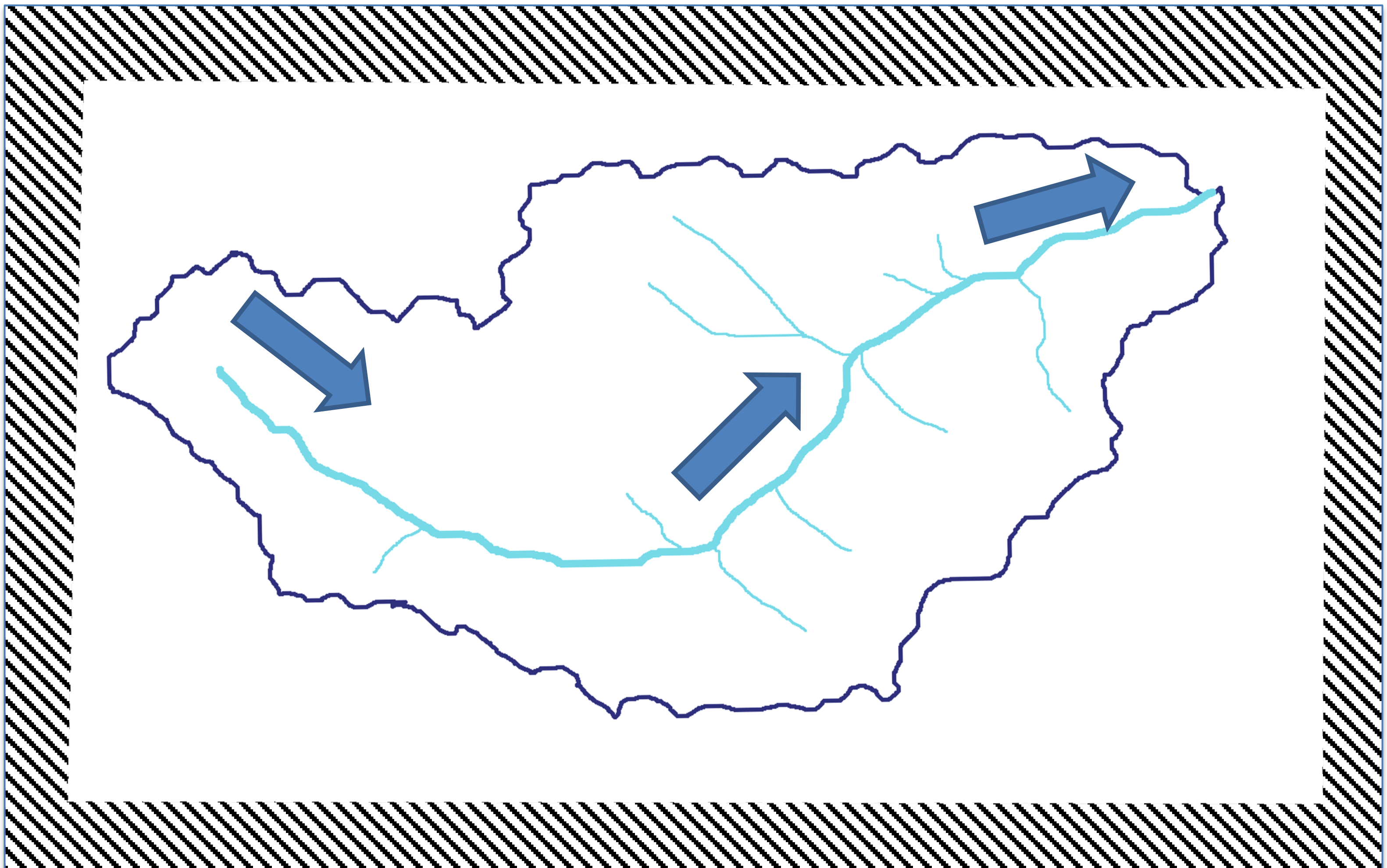
Note the z-upper is synched to computational grid, and is not linked with the Z multipliers

Boundary Conditions

- Boundary conditions must be set for all of the external patches on the domain
- 9 types of pressure boundary conditions (*manual 6.25*):
  1. DirEquilRefPatch: pressure will be in hydrostatic equilibrium with reference pressure
  2. DirEquilPLinear: pressure will be in hydrostatic equilibrium with piecwise line at elevation z=0
  3. FluxConst: Constant flux (L/T) normal to the patch
  4. FluxVolumetric: Constant volumetric flux normal to the patch
  5. Pressure File: Defines hydraulic head boundary conditions. Only the values on the specified patch will be used
  6. FluxFile: Flux boundaries read in from a pfb file
  7. OverlandFlow: Turns on fully-coupled overland flow routing for uniform fluxes (e.g. rainfall or ET over the entire domain)
  8. OverlandFlowPFB: Turns on fully-coupled overland flow routing for uniform fluxes with grid based spatially variable fluxes read in from a pfb
  9. Exact Solution: Exact know solution applied as a Dirichlet boundary condition on the patch
- Internal Dirichlet boundary conditions can also be defined by setting the pressure at internal points in the domain (*see the manual section 6.24*)

Boundary Conditions

- no-flow along all sides of the model domain



Boundary Conditions

```
model.BCPressure.PatchNames = "top bottom side"

model.Patch.bottom.BCPressure.Type = "FluxConst"
model.Patch.bottom.BCPressure.Cycle = "constant"
model.Patch.bottom.BCPressure.alltime.Value = 0.0

model.Patch.side.BCPressure.Type = "FluxConst"
model.Patch.side.BCPressure.Cycle = "constant"
model.Patch.side.BCPressure.alltime.Value = 0.0

model.Patch.top.BCPressure.Type = "OverlandKinematic"
model.Patch.top.BCPressure.Cycle = "rainrec"
model.Patch.top.BCPressure.rain.Value = -0.05
model.Patch.top.BCPressure.rec.Value = 0.0
```

3. Processing Topography

- Calculate slopes from elevations
- Adjust values to ensure a realistic drainage network

Processing topography

- There are built in tools to calculate slopes and do some processing
- Refer to the *Manual Section 8.2* for details
- For more advanced processing you can use:
  - Priority Flodd Tool (https://github.com/lecondon/PriorityFlow)
  - GIS tools *(See GRASS example on ParFlow blog, http://parflow.blogspot.com/2015/08/terrain-processing.html)*

```
set       dem         [pfload -sa dem.format_header.txt]
pfsetgrid {41 41 1} {0.0 0.0 0.0} {1000 1000 1.0} $dem

# Fill flat areas (if any)
# this routine interpolates across the bounds of flat areas to ensure nonzero slopes)
set       flatfill    [pffillflats $dem]

# Pitfill
# (this routine uses a standard pit-fill method to remove local minima
set       pitfill     [pfpitfilldem $flatfill 0.01 10000]

# Slopes
# (uses 1st-order upwind differences, consistent with PF overland flow scheme)
set       slope_x     [pfslopex $pitfill]
set       slope_y     [pfslopey $pitfill]

# PFB (slopes only...needed as PFB for parflow input)
pfsave $slope_x  -pfb  klam.slope_x.pfb
pfsave $slope_y  -pfb  klam.slope_y.pfb
```

Parking lot test

- Make your domain impervious, rain on it and look at the drainage network

Parking lot test

1. Set the permeability very low

```
model.Geom.Perm.Names = "domain"
model.Geom.domain.Perm.Type = "Constant"
model.Geom.domain.Perm.Value = 0.000001
```

2. Rain on the domain intermittently

```
#setup the timing of the cycles
model.Cycle.Names ="constant rainrec”

model.Cycle.rainrec.Names = "rain rec"
model.Cycle.rainrec.rain.Length = 5
model.Cycle.rainrec.rec.Length = 20
model.Cycle.rainrec.Repeat = -1

#Setup the rainfall rate
model.Patch.top.BCPressure.Type = "OverlandKinematic"
model.Patch.top.BCPressure.Cycle = "rainrec"
model.Patch.top.BCPressure.rain.Value = -0.05
model.Patch.top.BCPressure.rec.Value = 0.0
```

Note that the rainfall value is negative to indicate a flux in the negative z direction and units are L/T

Parking lot test

1. Set the permeability very low

```
model.Geom.Perm.Names = "domain"
model.Geom.domain.Perm.Type = "Constant"
model.Geom.domain.Perm.Value = 0.000001
```

2. Apply constant rain

```
#setup a time cycle for length of the simulation
model.Cycle.Names ="constant”

model.Cycle.constant.Names = "alltime"
model.Cycle.constant.alltime.Length = 1
model.Cycle.constant.Repeat = -1

#Setup the rainfall rate
model.Patch.top.BCPressure.Cycle = ”constant”
model.Patch.top.BCPressure.constant.Value = -2.1E-05
```

What if it won’t run?

1. Check that ParFlow is actually installed correctly and that you are able to run the test problems.
2. Look at the out.txt file to see if you are missing a key in your tcl script
3. Make sure that all of your input files are where they should be
4. If you are running on multiple processors make sure that every input file is being distributed and that the slopes files are distributed with NZ=1
5. Use PFTools to convert your inputs to silo and look to make sure they aren’t corrupted and the dimensions are right

http://*parflow.blogspot.com/2015/08/troubleshooting-models-that-dont-run-at.html*