ParFlow Tutorial

Reed Maxwell

Updated May 2016

ParFlow Equations: Steady Flow

$$\nabla \cdot \mathrm{q} = q_s$$

$$\mathrm{q} = -k(x)\nabla(\psi_p - z)$$

Groundwater flow and Darcy's Law

Pressure Head (L)

Saturated Hydraulic Conductivity (L/T)

ParFlow Equations: Richards' and Overland Flow

$$S_s S_w \frac{\partial \psi_p}{\partial t} + \phi \frac{\partial S_w(\psi_p)}{\partial t} = \nabla \cdot \mathrm{q} + q_s$$

$$\boldsymbol{q} = -k(x) k_r(\psi_p) \nabla(\psi_p - z)$$

Richards' Equation

$$\frac{\partial \psi_s}{\partial t} = \nabla \vec{\mathrm{v}} \psi_s + q_r$$

$$\boldsymbol{v}_x = -\frac{\sqrt{S_{f,x}}}{n} \psi_s^{2/3}$$

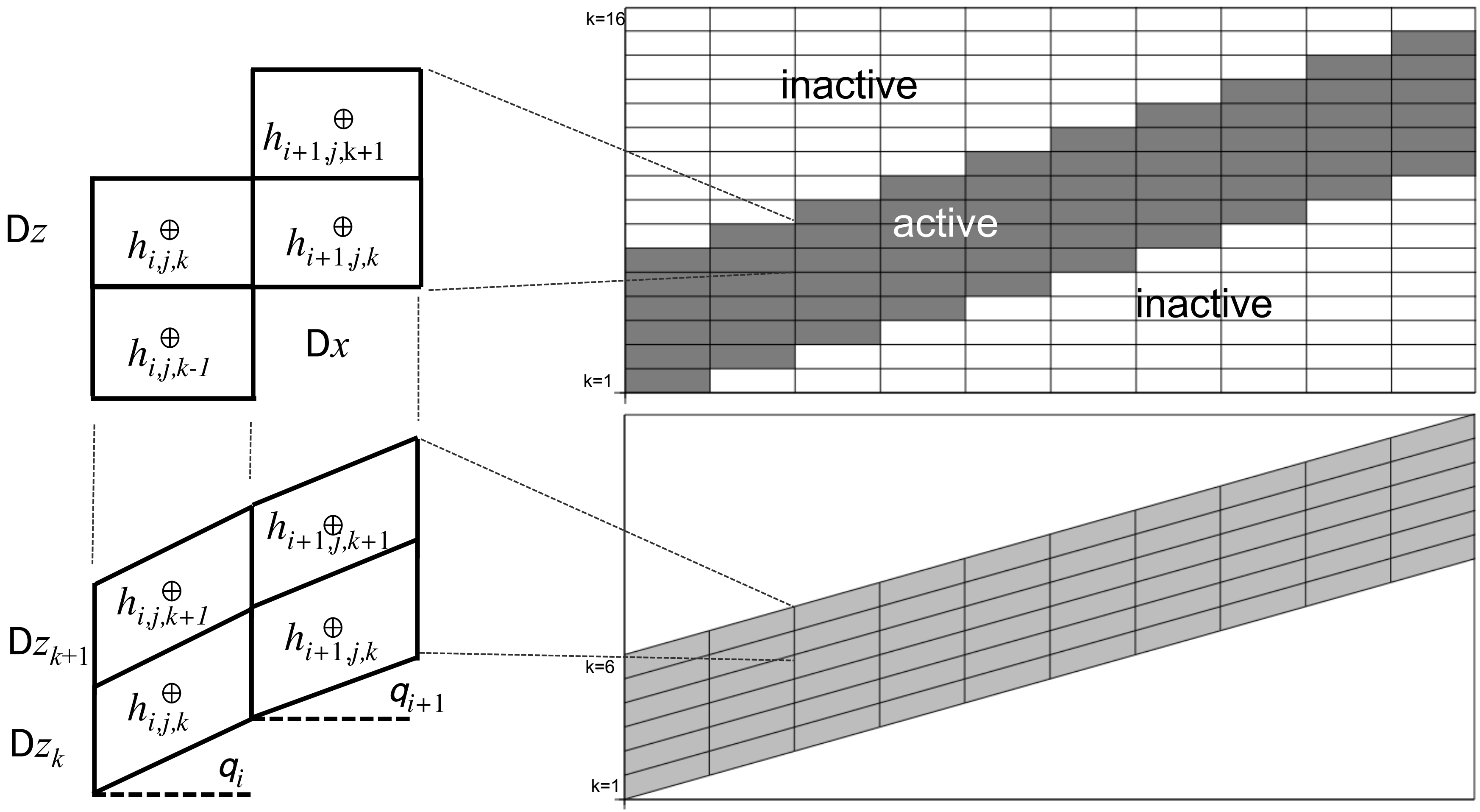
Kinematic Wave Equation

Pressure Head (L)

Saturation (-)

Saturated Hydraulic Conductivity (L/T)

ParFlow Gridding



Terrain Following Grid EQ

Modified Darcy's Law:

$$\mathbf{q} = -\mathbf{K}_s(\mathbf{x})k_r(h)[\nabla(h+z)\cos\theta_x + \sin\theta_x]$$

Slopes and fluxes:

$$\theta_x = \tan^{-1}(S_{0,x}) \;\; \text{and} \;\; \theta_y = \tan^{-1}(S_{0,y})$$

$$q_x = -K_{s,x}(\mathbf{x})k_r(h)\left[\frac{\partial(h)}{\partial x}\cos\theta_x + \sin\theta_x\right]$$

$$= -K_{s,x}(x)k_r(h)\frac{\partial(h)}{\partial x}\cos\theta_x - K_{s,x}(x)k_r(h)\sin\theta_x$$

Diffusive Pressure Term Topographic Term

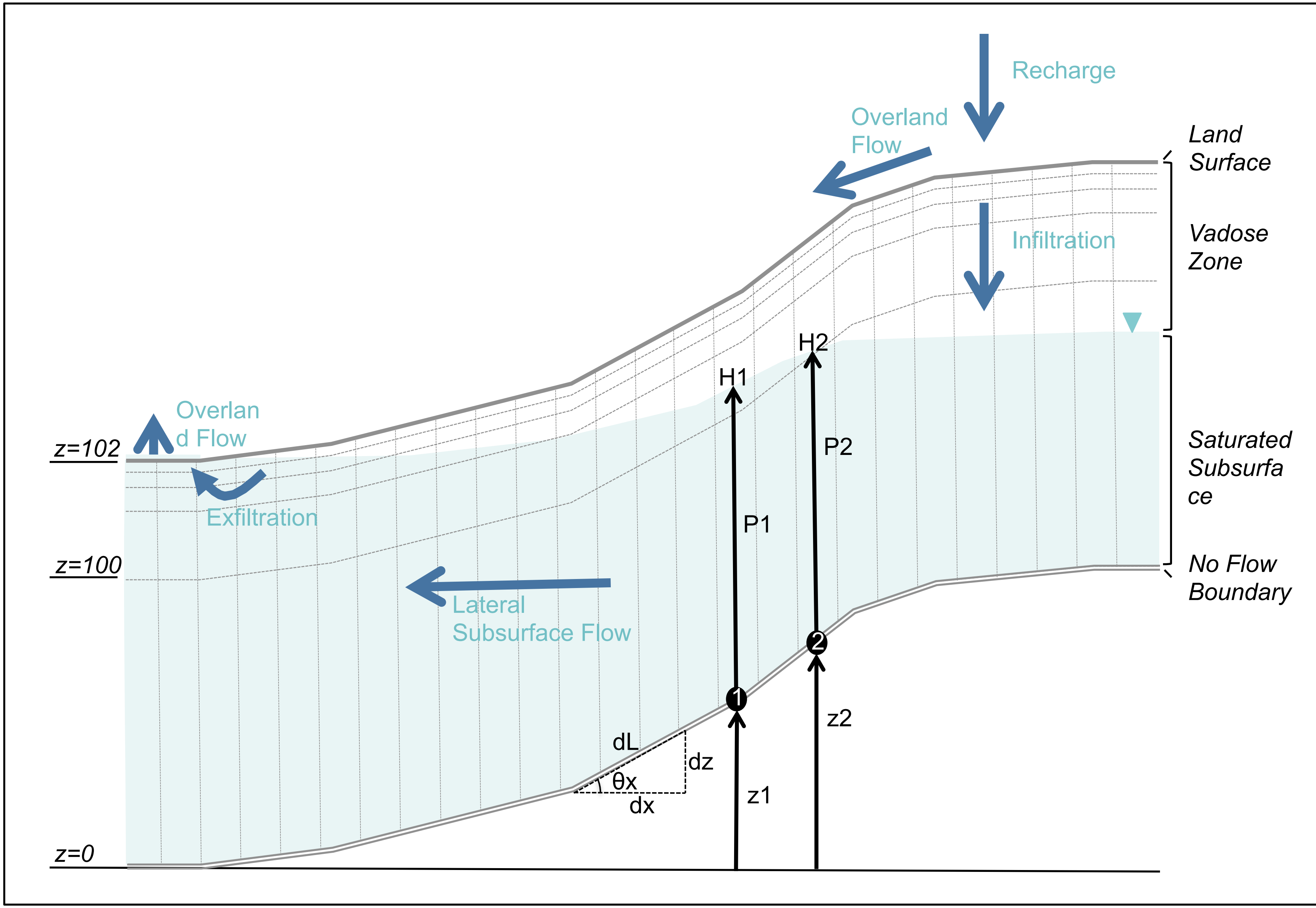


Figure 1: Illustration of a ParFlow model for an idealized hillslope using the terrain following grid formulation

Problem Definition

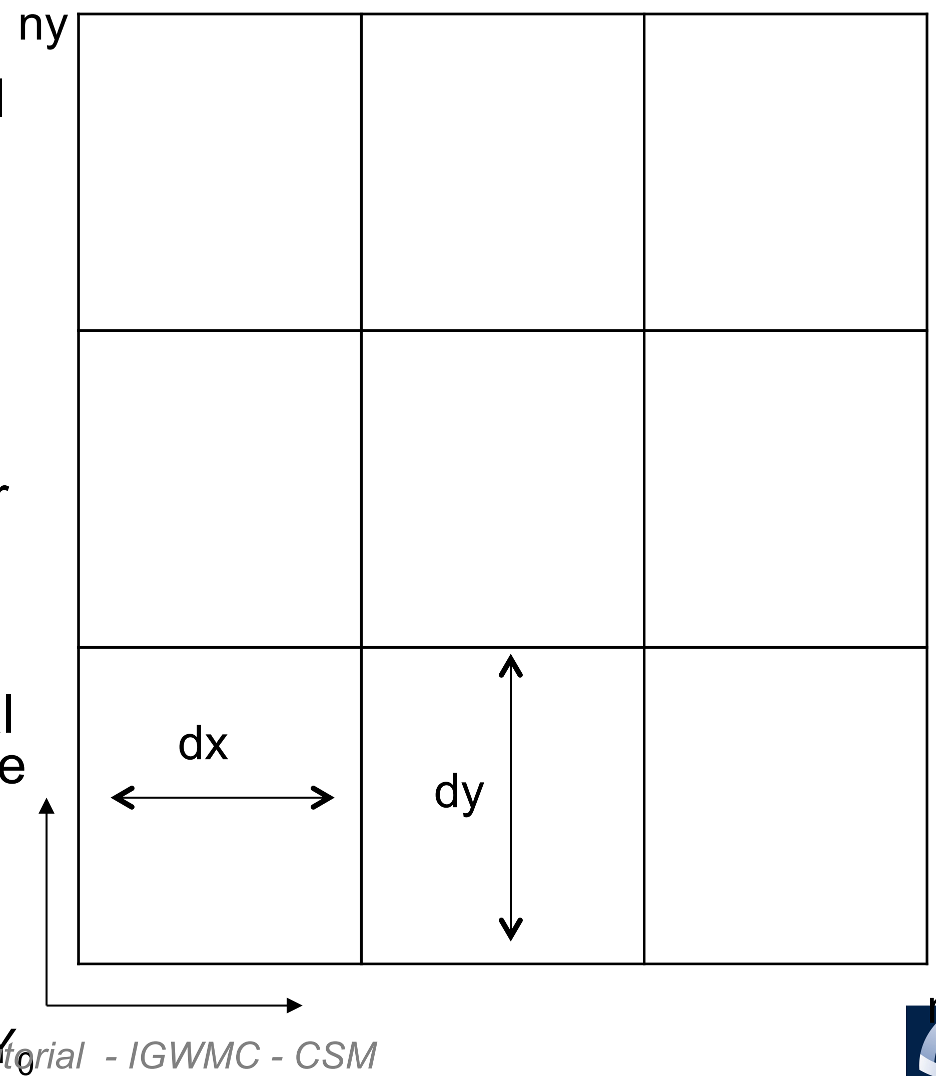
- Computational Grid
- Geometries
- Domain
- Parameters
  - Permeability
  - Porosity
  - Specific storage
  - Relperm
  - Saturation
  - Phases
  - Gravity
  - Toposlopes
  - Mannings coefficient
- Timing
  - Time steps
  - Time cycles
- Boundary Conditions
- Initial Conditions

Input File

- TCL/TK scripting language
- All parameters input as keys using `pfset` command
- Keys used to build a database that ParFlow uses
- ParFlow executed by `pfrun` command
- Since input file is a script may be run like a program

Computational Grid (§6.1.3)

- computational grid is a box that is the global outer shell of the problem
- it is defined by:
  - a lower x,y,z coordinate
  - cell diminsions (dx,dy,dz)
  - number of cells in each dimension (nx,ny,nz)
- grid spacing is uniform over problem
- though cubic the problem domain which defines the actual, active computational domain can be of any shape
- Code is cell-centered



Computational Grid (Input File)

Comment character for tcl/tk

```
#-----------------------------------------------------------
# Computational Grid
#-----------------------------------------------------------
pfset ComputationalGrid.Lower.X              0.0
pfset ComputationalGrid.Lower.Y              0.0
pfset ComputationalGrid.Lower.Z              0.0

pfset ComputationalGrid.NX                   30
pfset ComputationalGrid.NY                   30
pfset ComputationalGrid.NZ                   30

pfset ComputationalGrid.DX         10.0
pfset ComputationalGrid.DY                 10.0
pfset ComputationalGrid.DZ         .05
```

Coordinates (length units)

Grid dimensions (integer)

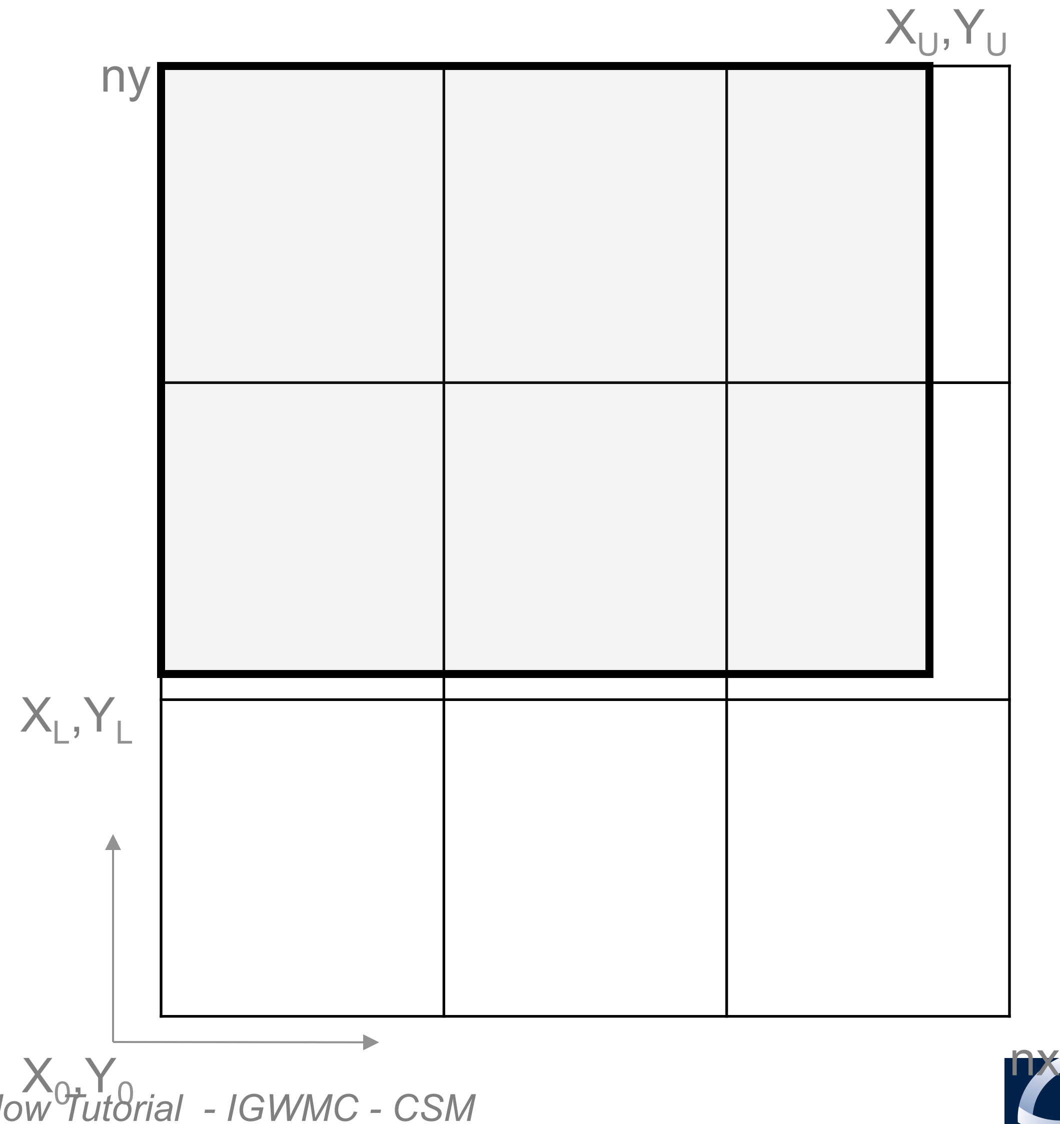
Cell size (length units)

Geometries (§6.1.4)

- Geometries are shapes that define aspects of the problem
- Any number is possible
- Combinations are fine
- Three types
  - Box
  - SolidFile
  - IndicatorField

Box Geometry

- a rectangular shape, specified within ParFlow input as upper and lower corner coordinates



Box Geometry (Input File)

```
#-----------------------------------------------------------
# The Names of the GeomInputs
#-----------------------------------------------------------
pfset GeomInput.Names                        "channelinput"

pfset GeomInput.channelinput.GeomName  channel

pfset GeomInput.channelinput.InputType  Box
#-----------------------------------------------------------
# Channel Geometry
#-----------------------------------------------------------
pfset Geom.channel.Lower.X                   140.0
pfset Geom.channel.Lower.Y                     0.0
pfset Geom.channel.Lower.Z                     0.0

pfset Geom.channel.Upper.X                    160.0
pfset Geom.channel.Upper.Y                    300.0
pfset Geom.channel.Upper.Z                      1.5
```

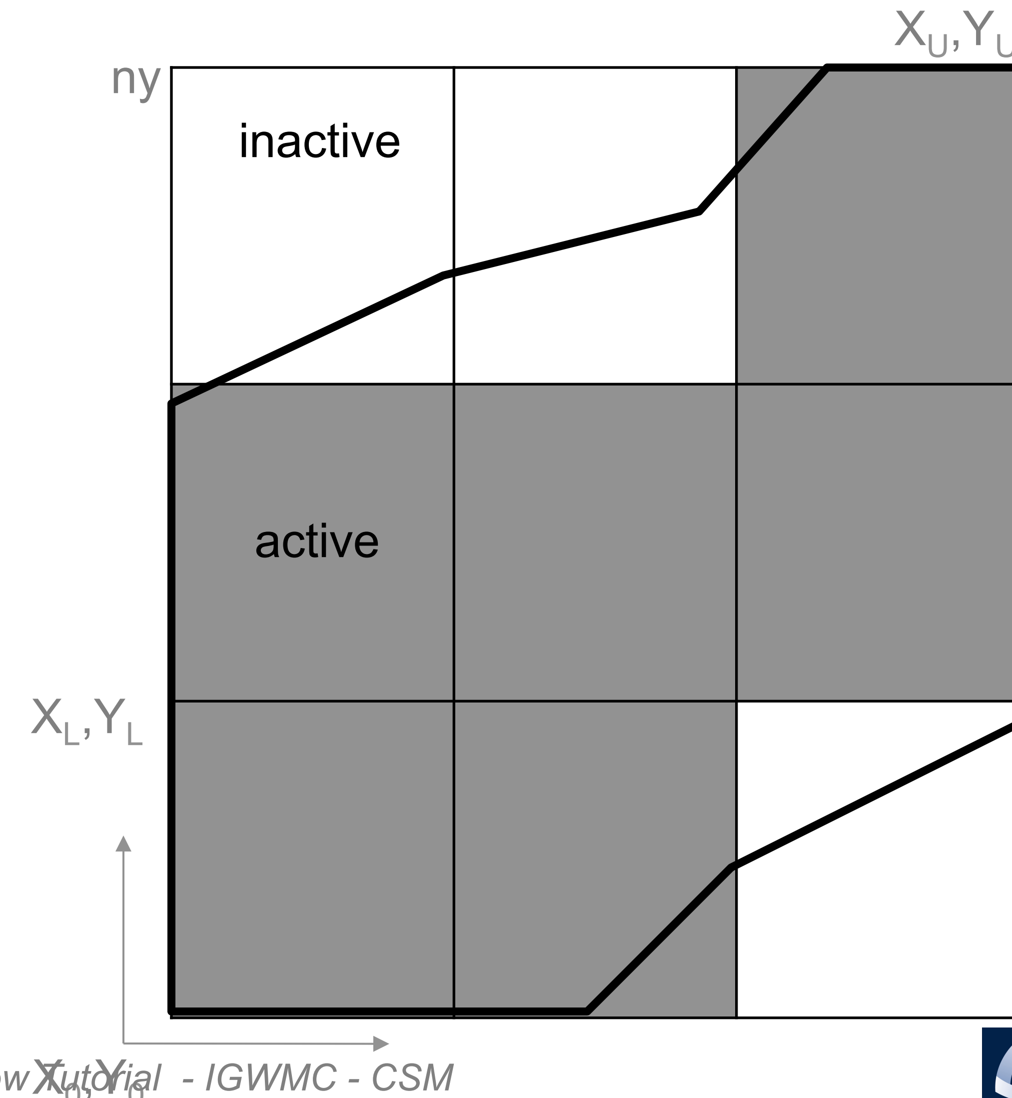
First define names for geometry inputs

Lower Coordinates (length units)

Upper Coordinates (length units)

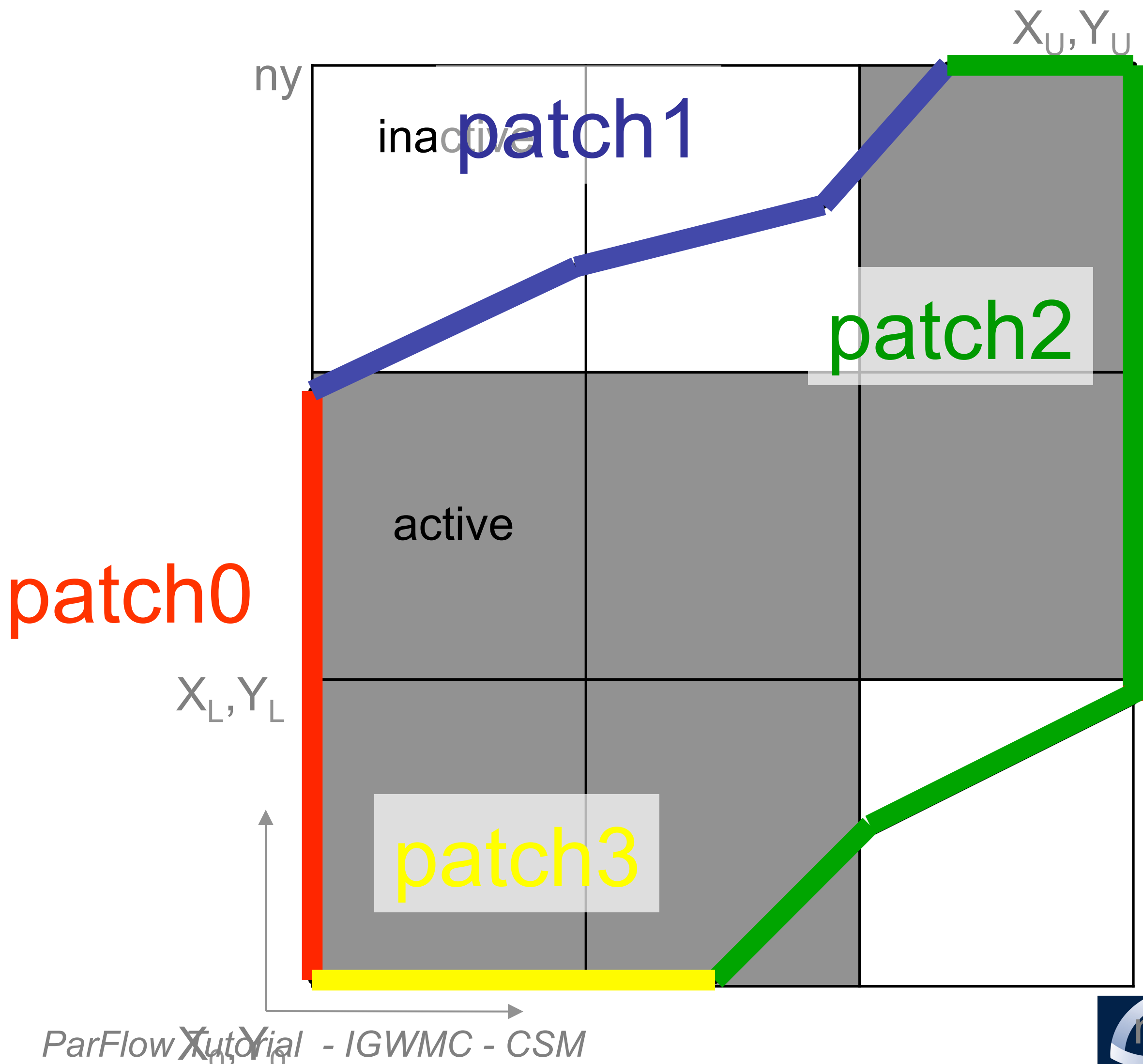
SolidFile Geometry

- A triangulated information network file that can delineate geometries of any shape
- Read in as a .pfsol file
- Geometries and patches are defined from within the file
- May be used to delineate active and inactive cells

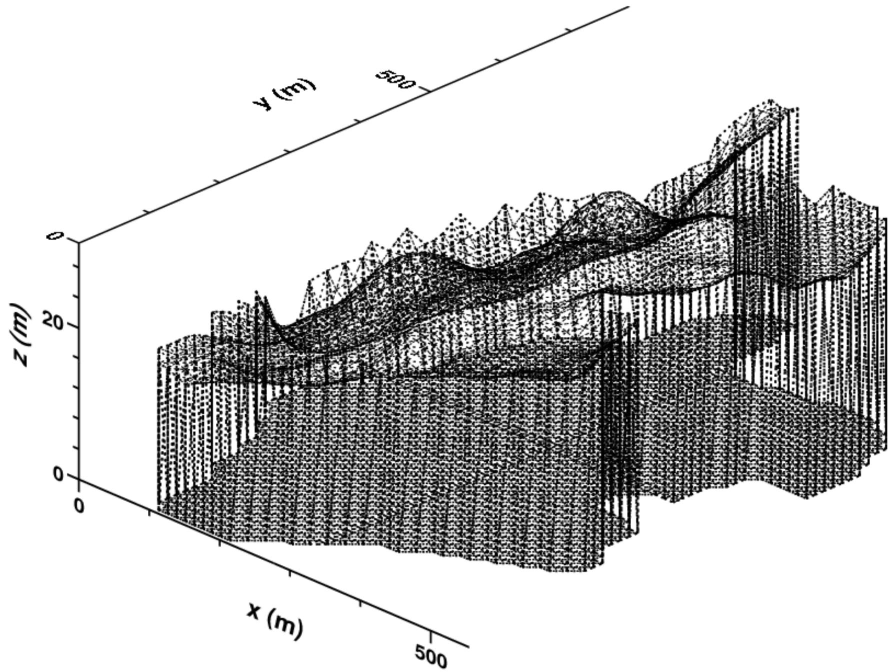


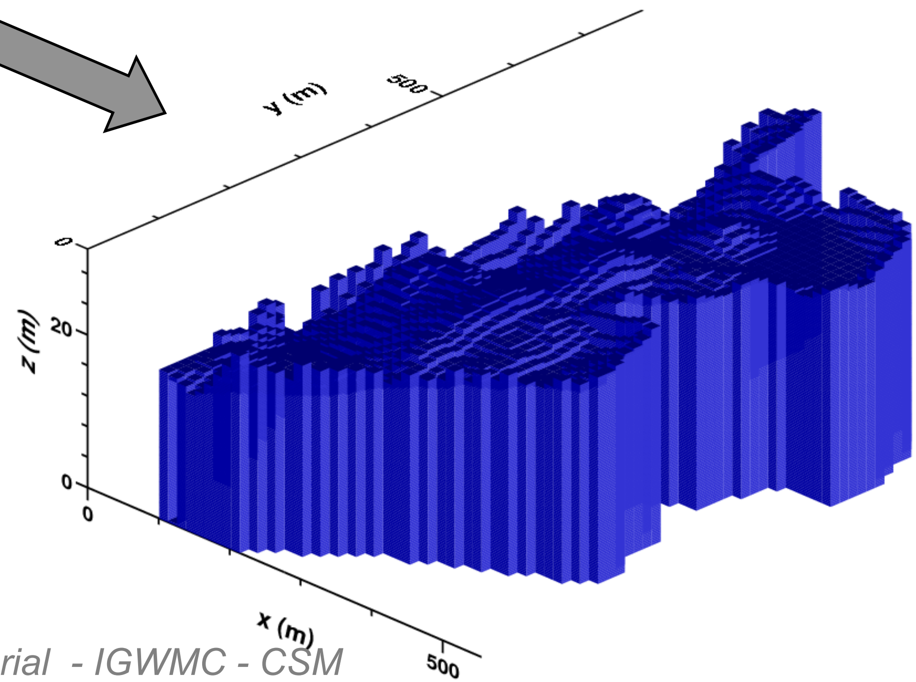
SolidFile Geometry- Patches

- patches can be any number or combination
- Must completely enclose geometry



SolidFile Geometry





SolidFile Geometry (input file)

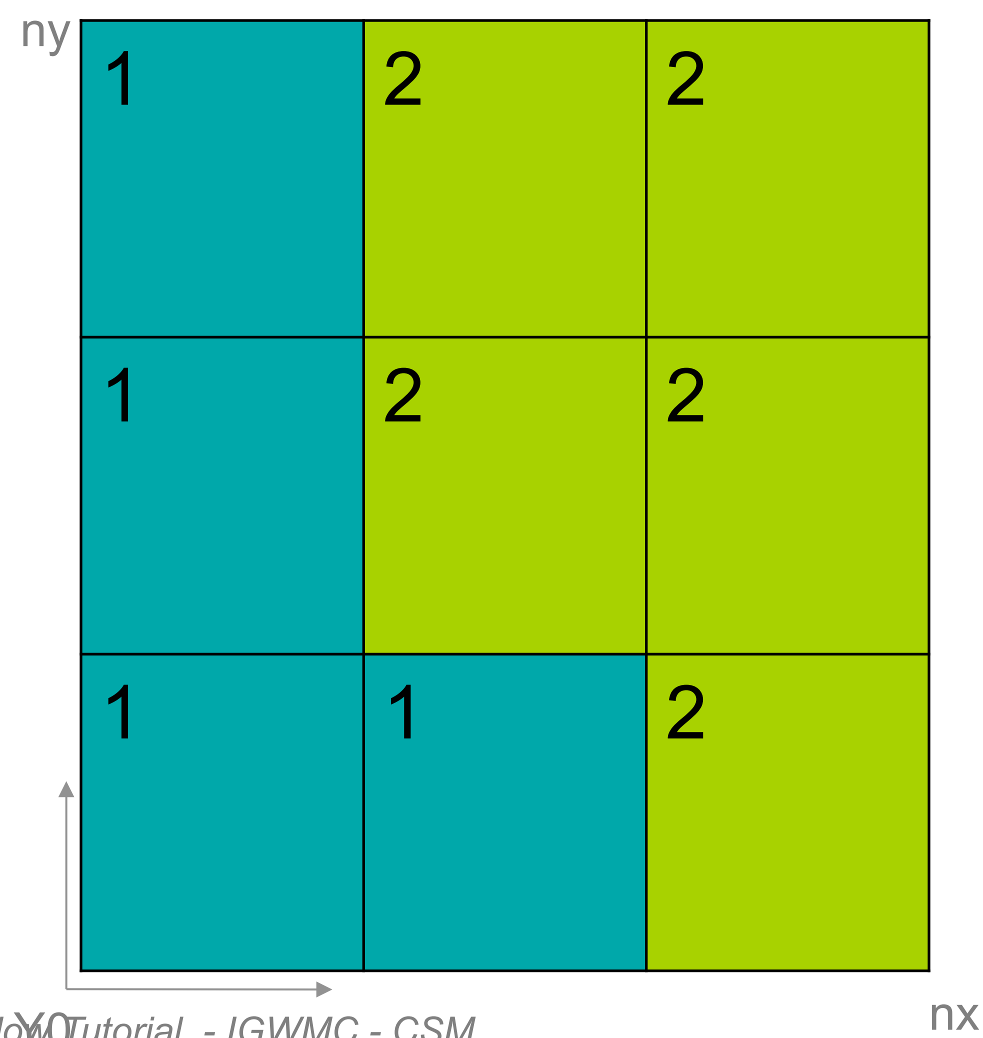
```
pfset GeomInput.Names                         "solidinput"

pfset GeomInput.solidinput.InputType  SolidFile
pfset GeomInput.solidinput.GeomNames  domain
pfset GeomInput.solidinput.FileName   fors2_hf.pfsol

pfset Geom.domain.Patches                     "infiltration z-upper
     x-lower y-lower x-upper y-upper z-lower"
```

IndicatorField

- an indicator field (.pfb file) with integer numbers for every cell in the computational domain.
- These are mapped to geometry names within the parflow .tcl input file



IndicatorField (input file)

```
#-----------------------------------------------------------
# Indicator Geometry Input
#-----------------------------------------------------------
pfset GeomInput.indi_input.InputType            IndicatorField
pfset GeomInput.indi_input.GeomNames             "F1 F2 F3"
pfset Geom.indi_input.FileName   "example.pfb"

#f1 = background
#f2 = sand ARPS 1
#f3 = sandy loam, arps 3

pfset GeomInput.F1.Value                1
pfset GeomInput.F2.Value                2
pfset GeomInput.F3.Value                3
```

Integer numbers in file mapped to named keys

Domain (§6.1.7)

- one of the geometries has to be specified as the outer domain for the problem
- this has to be either a Box or a SolidFile type as patches covering the entire exterior of this geometry (§.5.1.5 p.51) need to be defined to assign boundary conditions (§. 5.1.21-22) to the simulation
- this geometry should be the same size, or smaller, than the computational grid

Domain (input file)

```
#-----------------------------------------------------------------
# Domain
#-----------------------------------------------------------------

pfset Domain.GeomName test
```

Geometry test must be box of solidfile, must have patches that can be used to assign boundary conditions

Permeability (§6.1.11)

$$\boldsymbol{q} = -k(x) k_r(\psi_p) \nabla(\psi_p - z)$$

- Hydraulic conductivity what usually used (L/T)
- Sets length an time units for entire problem
- Intrinsic permeability equates to hydraulic conductivity for density=gravity=viscosity=1 (otherwise units L²)

$$k(x) = \frac{\kappa \rho g}{\mu}$$

- Can be spatially-heterogeneous
- Tensor

Permeability (input file)

```
#-----------------------------
# Perm
#-----------------------------

pfset Geom.Perm.Names                       "left right channel"

# Values in m/hour
pfset Geom.channel.Perm.Type                  Constant
pfset Geom.channel.Perm.Value                 0.00001

pfset Perm.TensorType                     TensorByGeom

pfset Geom.Perm.TensorByGeom.Names  "domain"

pfset Geom.domain.Perm.TensorValX  1.0d0
pfset Geom.domain.Perm.TensorValY  1.0d0
pfset Geom.domain.Perm.TensorValZ  1.0d0
```

First define names for geometry inputs

Type and value (L/T) or (L^2)

Principle axis tensor multipliers (-)

Permeability (input file, ex2)

```
#---------------------
# Perm
#---------------------
pfset Geom.Perm.Names                                "left right channel"
# Values in m/hour
# these are examples to make the upper portions
# of the v heterogeneous
pfset Geom.left.Perm.Type "TurnBands"
pfset Geom.left.Perm.LambdaX  50.
pfset Geom.left.Perm.LambdaY  50.
pfset Geom.left.Perm.LambdaZ  0.5
pfset Geom.left.Perm.GeomMean  0.01

pfset Geom.left.Perm.Sigma   0.5
pfset Geom.left.Perm.NumLines 40
pfset Geom.left.Perm.RZeta  5.0
pfset Geom.left.Perm.KMax  100.0
pfset Geom.left.Perm.DelK  0.2
pfset Geom.left.Perm.Seed  33333
pfset Geom.left.Perm.LogNormal Log
pfset Geom.left.Perm.StratType Bottom
```

Parallel turning band implementation follows *Tompson et al (1989) WRR 25(10):2227–2243*

Correlation lengths (L) and K_g of K_{sat} (L/T,L^2)

s_{lnk} (not s_{lnk}^2)

Numerical parameters for TB code, discussed in detail in Tompson (1989), seed and lines most important

Type of distribution and whether it follows geometry

Porosity and Specific Storage (§6.1.12, 6.1.13)

```
#-----------------------------------------------
# Specific Storage
#-----------------------------------------------
pfset SpecificStorage.Type            Constant
pfset SpecificStorage.GeomNames       "domain"
pfset Geom.domain.SpecificStorage.Value 1.0e-4

#-----------------------------------------------
# Porosity
#-----------------------------------------------
pfset Geom.Porosity.GeomNames           “domain"

pfset Geom.domain.Porosity.Type           Constant
pfset Geom.domain.Porosity.Value          0.25
```

Relative Permeability (§6.1.19) and Saturation (§6.1.22)

- May be **Constant** or use **VanGenuchten**, **Haverkamp**, or **polynomial** functions
- May vary parameters with geometry but same type (e.g. VanGenuchten) for entire domain
- May read parameters in from a .pfb file for VanGenuchten (for every cell in computational domain)

Timing (§6.1.5)

- Timing only used for solver Richards
- Time steps may be constant or variable
- Timing section provides link between time steps and time cycles (next)
- Time units set by k, K units as described previously

Timing (§6.1.5)

```
#-----------------------------------------------------------
# Setup timing info
#-----------------------------------------------------------
pfset TimingInfo.BaseUnit          1.0
pfset TimingInfo.StartCount        0
pfset TimingInfo.StartTime 0.0
pfset TimingInfo.StopTime          300.0
pfset TimingInfo.DumpInterval      30.0
pfset TimeStep.Type         Constant
pfset TimeStep.Value               10.0
```

Sets time units for time cycles (T)

Initial output file number

Start and finish time for simulation (T)

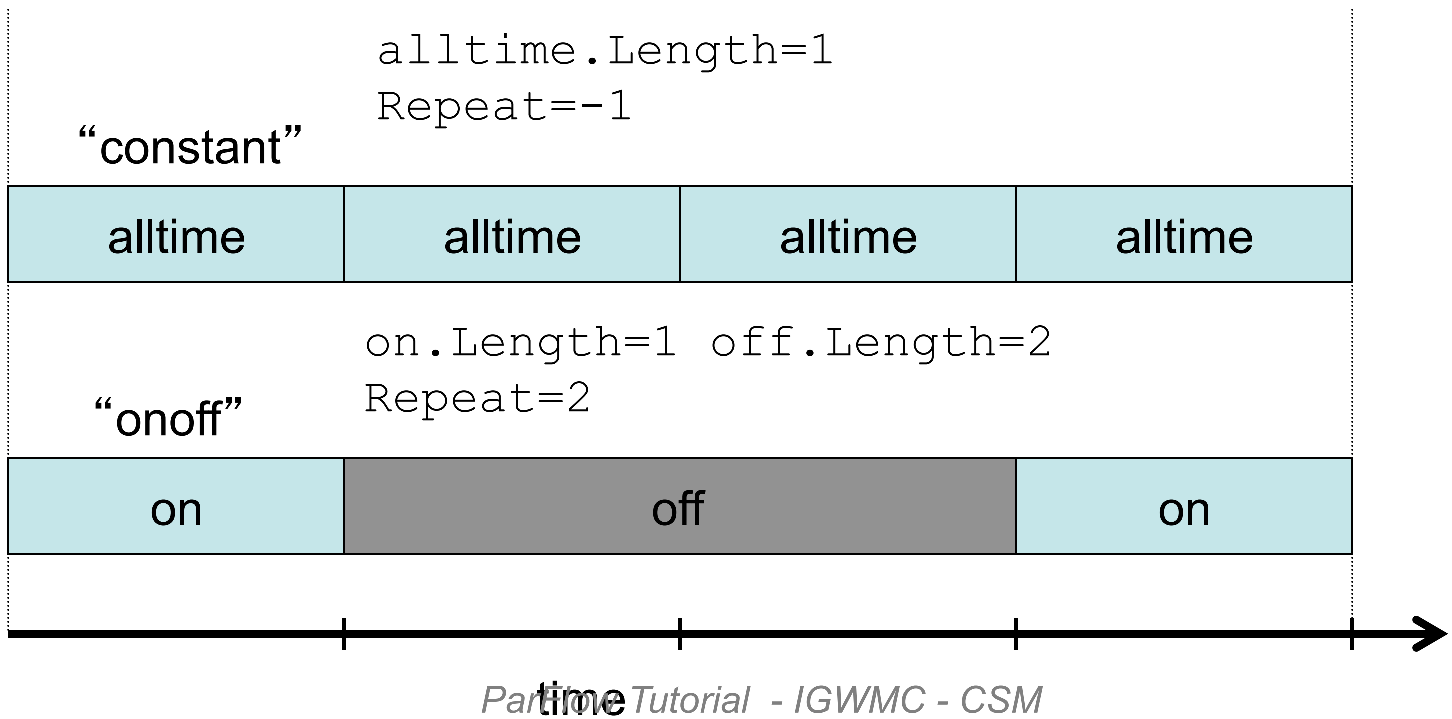
Interval to write output (T)
-1 outputs at every timestep

Timestep type

DT (T)

Time Cycles (§6.1.6)

- Time cycles are named lists
- All cycles are *integer multipliers* of `BaseUnit` value defined previously
- May be used for BC's and wells.



Time Cycles (§6.1.6)

```
#-----------------------------------------------------------
# Time Cycles
#-----------------------------------------------------------
pfset Cycle.Names                   "constant onoff”

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              1
pfset Cycle.onoff.off.Length             2
pfset Cycle.onoff.Repeat                 2
```

Length of time cycle and repeat value

Length of each time cycle and repeat value

Boundary Conditions (§6.1.24)

- Applied to patches defined by domain
- Many BC Types, two categories
  - Dirichlet
  - Flux
- Dirichlet Equilibrium sets a *constant head potential* by setting hydrostatic *pressure head*
- Assigned for each patch in the `Domain` geometry
- BC Time Cycles

Boundary Conditions (§6.1.24)

```
#-----------------------------------------------------------
# Boundary Conditions: Pressure
#-----------------------------------------------------------
pfset BCPressure.PatchNames “X0 Xmax Y0 …"
pfset Patch.X0.BCPressure.Type              DirEquilRefPatch
pfset Patch.X0.BCPressure.Cycle             "constant"
pfset Patch.X0.BCPressure.RefGeom domain
pfset Patch.X0.BCPressure.RefPatch          bottom
pfset Patch.X0.BCPressure.alltime.Value      10.0

pfset Patch.Xmax.BCPressure.Type              DirEquilRefPatch
pfset Patch.Xmax.BCPressure.Cycle   "constant"
pfset Patch.Xmax.BCPressure.RefGeom           domain
pfset Patch.Xmax.BCPressure.RefPatch          bottom
pfset Patch.Xmax.BCPressure.alltime.Value 9.97501

pfset Patch.Y0.BCPressure.Type              FluxConst
pfset Patch.Y0.BCPressure.Cycle             "constant"
pfset Patch.Y0.BCPressure.alltime.Value 0.0
```

Pressure value at reference patch

Flux value (L/T)

Wells (§6.1.30)

- Wells defined as a list of names
- Wells are located by X,Y, Z_{top} and Z_{bottom} coordinates, then snapped to grid
- Wells can be "vertical" or "recirculating"
- Wells can be specified by a pressure or a flux
- Wells fluxes can be equally divided over the screen or weighted by K

Wells (§6.1.30)

```
pfset Wells.Names                       "rnm2s"

pfset Wells.rnm2s.InputType             Vertical

pfset Wells.rnm2s.Cycle                 constant

pfset Wells.rnm2s.Action                Extraction
pfset Wells.rnm2s.Type                  Flux

pfset Wells.rnm2s.X            3548.
pfset Wells.rnm2s.Y            1893.
pfset Wells.rnm2s.ZLower                616.
pfset Wells.rnm2s.ZUpper                640.

pfset Wells.rnm2s.Method                Weighted

pfset Wells.rnm2s.alltime.Flux.water.Value  1635.298
```

Flux-type well and we are extracting*

X,Y, Z Coordinates of the well (L)

Flux is weighted by K

Extraction flux (L^3/T)*

Running ParFlow

- Parallelization
- Distributing files
- `pfrun`
- Undistributing files
- Manipulating and viewing output

Parallelization

- Domain parallelized by specifying number of processor divisions in x,y,z
- Parallelization done on computational domain
- Done using P,Q,R values
  - Total processors=P*Q*R
  - Domain divided by nx/P, ny/Q, nz/R
- Load balancing issues

Parallelization (input file)

```
pfset Process.Topology.P 1
pfset Process.Topology.Q 1
pfset Process.Topology.R 1
```

Single processor simulation,
P,Q,R are integer values

```
pfset Process.Topology.P 4
pfset Process.Topology.Q 2
pfset Process.Topology.R 1
```

Eight processor simulation,
P*Q*R=4*2*1=8

Distributing Files (§4.2)

- ParFlow reads and writes parallel files
- One portion of the file per processor (except for sequential/shared memory build)
- ParFlow binary files (.pfb) must be distributed (split up) before being read in
- ParFlow binary files (.pfb) must be undistributed at the end of the simulation
- Two tools to do this, `pfdist` and `pfundist`, may be run directly in tcl input script.

Distributing Files (input file)

```
pfdist my.input.file.pfb
```

Distribute an input file
Must have specified processor topology, can happen anywhere in script before pfrun command

```
pfundist default_over

pfundist my.input.file.pfb
```

First line undistributes an entire run

Second line undistributes a particular file

Distributing Files (file structure)

	Undistributed	`pfdist` `(P=2,Q=1,R=1)`	Distributed
MPI	`my.input.file.pfb`	→	`my.input.file.pfb.00000` `my.input.file.pfb.00001`
		`pfundist`	
MPI	`my.input.file.pfb`	←	`my.input.file.pfb.00000` `my.input.file.pfb.00001`

Running ParFlow

- `pfrun` command
  - Builds database of keys
  - Executes program
- Some error checking of keys
- Actual command line runs executable or mpirun's executable
- May run parflow code more than once in single script
- Need parflow package/header information

Running ParFlow (input file)

```
# Import the ParFlow TCL package

lappend auto_path $env(PARFLOW_DIR)/bin
package require parflow
namespace import Parflow::*

…
pfrun myrun
```

Load the parflow tcl package, no pf commands work without this

Run parflow, project name is myrun, this dictates all output names

Running Parflow (how it works)

TCL input script:
Set database keys for simulation, any other manipulations.

`pfrun` command:

1. Executes `parflow.tcl` script
2. Write database (`.pfidb`) file
3. Set up parallel run parameters
4. Execute `run` script

`run` script:

1. Execute ParFlow using platform specific options
2. Port standard output to a file

Running ParFlow (file structure)

- Project name is the base for all output
- Most output is `*project*.out.*var*.*time*.ext`

For a project called 'myrun'

Log files:

```
myrun.out.log
myrun.out.kinsol.log
```

Perm/porosity files:

```
myrun.out.perm_x.pfb
myrun.out.porosity.pfb
```

Pressure/Saturation files:

```
myrun.out.press.00001.pfb
myrun.out.satur.00001.pfb
```

Output time step, 00000 is initial, integer values depending on output times

Mask file:

```
myrun.out.mask.00000.pfb
```

The mask is a file of zero's and ones, 0=inactive cell, 1=active cell

Other/diagnostic files:

`myrun.pfidb` Parflow database

`myrun.out.pftcl`

`myrun.out.txt` Line output

Output, viewing, manipulating

- Several options for output/visualization
- PFTools can be used to read/write and perform operations on output and input files

ParFlow File Types

- PFB: **P**ar**F**low **B**inary. ParFlow's native file type, can be written, read into ParFlow, read into and written by PFTools.
- SILO. VisIt's native file type, can be written by ParFlow, read into and written by PFTools

File Parallelism

- ParFlow has several options for parallel io
  - PFB may be distributed as *n* files or as a single file with companion file (.dist)
  - SILO has two options, PMPIO where *n* processors write to *m* files and regular where *n* files are written
- The best file type depends upon application

VisIt

- Free, developed at LLNL (http://www.llnl.gov/visit/)
- Powerful rendering tool
- Runs on multiplatform and parallel
- SILO format, which has many options within ParFlow (converting or IO), fully-supported

PFTools Commands (§4.2)

- Many commands load and write files
- `pfload` reads files that are parflow binary, simple binary and ascii
- `pfsave` writes files that are parflow binary, simple binary and ascii
- One a dataset is loaded (from a file) it may be manipulated with many different tools commands (e.g. convert pressure head to head potential)

PFTools Commands - Examples

```
pfrun myrun
pfundist myrun
```

Run parflow and undist output, project name is myrun, this dictates all output names

Loaded data is placed in tcl variable “press”, notice later we use this var as “$press”

```
set press [pfload myrun.out.press.pfb]
set head [pfhhead $press]
pfsave $head -pfb myrun.head.pfb
```

Load pressure output

Convert pressure to head, note the tcl vars

Save calculated head

Save as ParFlow binary file, -sa would be simple ascii, for example