ParFlow

Raphael Nussbaumer

June 30, 2014

Contents

Chapter 1

Introduction

This manual was written during my Master where I learnt to use ParFlow (which, by the way, mean: ???). I will try to focus on very simple and obvious manipulation for new-user and some subtle tricks that I found quiet useful.

1.1 What is ParFlow ?

Developped at Lawrence Livermore National Laboratory (LLNL), we can find on their official ParFlow website

> *ParFlow is an integrated, parallel watershed model that makes use of high-performance computing to simulate surface and subsurface fluid flow. The goal of the ParFlow project is to enable detailed simulations for use in the assessment and management of groundwater and surface water, to investigate system physics and feedbacks and to understand interactions at a range of scales.*

1.2 Some useful resources

Before reading this manual, I strongly recommend to read the following resources (special the first one):

- The Parflow user's manual (december, 2010) will become your bedside book...
- Professor Reed Maxwell website host the source file download link(look also at his presentations on the website),
- Parflow blog will help you to install ParFlow,
- The original parflow project website gives some papers using PARFLOW

- The mailing list can be used to ask questions
- And some other presentations :
  - The Baltimore exemple
  - ParFlow exemple
  -
  - http://www.cosmo-model.org/content/tasks/workGroups/wg3b/meetings/2012-soilveg/COSMO-CLM2-ParFlow.pdfCLM presentations

Chapter 2

Presentation of ParFlow

In this introduction, we will try not to speak about computer. The first think you want to know is how does parflow think.
The idea of his designer is that Hydrological processes can't be represented as lumped "box" but that the spatial et temporal variation of each processes need to be represented as a continuum. (eg: Richards' equation is used in saturated AND unsaturated). To do this, a grid approach is necessary and only possible because of major advance in the computer science field.

2.1 Two solvers

PARFLOW can run in two different mode:

- IMPES : steady state, fully saturated, single phase,
- RICHARDS : variably-saturated, transient conditions, overland flow and ability to be couple with CLM

As you can see, you certainly want to use Richards' solver for whatever you want to do...

2.2 TCL/TK

TCL stands for Tool Command Language, it is a script language which rather than compile, interpret and automate the execution of tasks that could alternatively be executed one-by-one by a human operator. (similar with a script in matlab ie:not a function)

Here is an exemple how to write TCL command in a file that has a '.tcl' extention

```
# this is a comment
set x 2
set y 3
set res [expr $x*$y]
puts "2 * 3 is $res."
```

To run the file juste use the tclsh command in bash

```
$tclsh script.tcl
```

You could also run line by line in the bash:

```
$ tclsh
% commandName argument1 argument2
%
% exit
```

Some rules:

- [ ]: return the result of evaluating the script contained inside.
- $var return the value of the variable var
- more info on how to write TCL script, command list and a tutorial.

You can also run TCL from matlab but on my computer I add to recreate the environment variable `PARFLOW_DIR` and then run the script with the bang command (!)

```
setenv('PARFLOW_DIR', '/home/raphael/parflow/parflow')
!tclsh post.tcl
```

2.3 Coupled

One big strength of ParFlow is his ability to use CLM (Common Land Model) in a fully coupled fashion (like a module). This means that for each time step, it solves only one big matrix (not like two separate with an exchange flux). It avoid boundary condition errors and make the mode energy and mass conservative. CLM is part of the Land Surface Model (LSM) family (like Jules). For info, Maxwell and Miller (2005) presents the first model (undistributed, not fully coupled) but the current version is the one describe in Kollet and Maxwell (2008).

It as also been coupled with the mesoscale atmospheric model ARPS. SLIM is a Lagrangian, particle-tracking that can also be install with ParFlow.

2.4 Multigrid

Fully parallel, multigrid-preconditioned, finite difference/finite volume 3D flow Multigrid method accelerate the iterative solution by solving a coarse version of the version and using that to inform the fine grid solution (Ashby and Falgout, 1996) (smooth to 1x1 solve it, then 2x2x2 solve it (interpolation)) . This is part of the preconditioner Parralelism: (Falgout and Jones (1999)) the numerical algorithm (solver) must be scale linearly with problem size Solver used is an Implicit method : Newton Methods Message passing distributed memmory

2.5 Which OS ?

It can't be install on Windows but both Mac and Linux should work. You might want to consider to use a virtual box otherwise. But I would suggest to install Ubuntu (THE user-friendly Linux distribution) in a dual boot. Matlab, Visit, Python... what ever you would need are also available on Ubuntu.

Some

- : home directory
- ./ : current directory
- ../: parent directory
- the [tab] button auto-completes your typing on the command line.

Chapter 3

Installation

3.1 What is compilation

What better than Wikipedia when you don't know something ?

> *A compiler is a program that transforms source code written in a programming language into another computer language. The most common reason for wanting to transform source code is to create an executable program.*

It means than when you install a software, in order to run, the code of the software needs to be translate so that your OS can understand it. On windows, the '.exe' does this job, but Linux has so many distributions that we would need that many executable. On Ubuntu, apt-get automaticaly retrieve, configure and install of software packages by compiling source code. The use of '.deb' is the extension of the Debian software package format.

A lot of file appear that are the ingredient for the meal... what you need to find is the recipe : './configure'. This is an important script that check for the presence of the library required for the installation as well as creating the make file.

```
$./configure
```

If you just run ./configure, it will check that you have everything or will tell you what you need... Once it tells you that you're fine, just compile with make.

```
$make
```

Once the compilation is finish run the executable with sudo make install

```
$sudo make install
```

As you will discover, compilation can be customize a lot with
http://stackoverflow.com/questions/10961439/why-always-configure-make-make-install-as-3-seperate-steps

3.2 How to install it

I used a lot what the blog prescribe with some additional indication or explanation.

First, as suggested by John build a directory structure for your ParFlow installation called parflow. in home (or). Place all the things you download there.

3.2.1 Download and install compilers

You need to install the tools to allows to compile with :

```
$sudo apt-get install build-essential
```

First, download and install the compilers (remember, the one that convert the source code to an executable for your machin) C, C++ and Fortran compilers and Tcl/Tk:

```
$sudo apt-get install gcc g++ gdc gfortran tcl-dev tk-dev
```

3.2.2 Download Libraries

- **Download OpenMPI** OpenMPI is a message-passing system which rather than directly invoke a function by name (as in conventional programming), MP sends a *message*. This powerfull tool allows separate program to work together. This is used when using parallel processor.
- **Download SILO**: SILO is a library for reading and writing a wide variety of scientific data to binary file. One of the more popular tools that process Silo data files is the VisIt visualization tool.
- **Download HYPRE**.HYPRE is a library for solving large, sparse linear systems of equations on massively parallel computers. ParFlow
- **Download ParFlow**, well you know who is PARFLOW

3.2.3 Install Libraries

All the downloaded files are archive which consits have been compress ('.gz') and combine ('.tar'). To extract in command line use:

```
$tar -xvf FileName.tar.gz
```

Here are two usefull idea of John: You can also rename all the folder without the version number for simplicity.

And add environmental variables to Linux to have easier path. You can create a sort of global variable that will be accesible from everywhere in the shell. To do so add the following line in the `~/.bash_aliases` file.Unix shell executes this file when it start (here some tips)

```
        $export CC=gcc
        $export CXX=g++
        $export FC=gfortran
        $export F77=gfortran

        $export PARFLOW_DIR=/home/john/parflow/parflow

        $export SILO_DIR=~/parflow/silo
        $export HYPRE_DIR=~/john/parflow/hypre
        $export PATH=$PATH:~/john/parflow/visit/bin
```

If you don't want to restart the shell, you can source the new file:

```
$source ~/.bash_aliases
```

Now, to install all the 4 libraries, you have to compile them as explain before.

Install OpenMPI

```
$cd openmpi
$sudo ./configure
$sudo make all
$sudo make check
$sudo make all install
$cd ..
```

Compare to our theoretical example, we use `make all`: this will compile not only the first makefile target but all of them. `make check` will runs post-build tests in the build directory.

Install Silo

```
$cd silo
$./configure --prefix=$SILO_DIR --disable-silex
$make all
$make check
$make all install
$cd ..
```

Install Hypre

```
$cd hypre/src
$./configure --prefix=$HYPRE_DIR --with-MPI [if you re not using MI
$make all
$make check
$make all install
$cd ../..
```

Install ParFlow and PFTools, run the test cases

Note: if youre not using MPI, leave the last flag off.

```
    $cd parflow/pfsimulator
    $./configure --prefix=$PARFLOW_DIR --enable-timing --with-clm --with-
    $make install
    $cd ../pftools
    $./configure --prefix=$PARFLOW_DIR --with-silo=$SILO_DIR --with-amps=
make install
    $cd ../test
    $sudo ldconfig
    $make check
```

More importantly, before continuing we need to modify the Makefile to fix a few errors or else the installation will fail. Within the "config" folder, there exists a file called "Makefile.config"; locate it and open it in the editor of your choice. The changes we need to make happen in the LDLIBS specification near the bottom of the file. There are three changes to make: 1) add a "-l" in front of "gfortran", 2) remove the stray "-l", and 3) remove the space between a "-l" and "gfortran". This line cannot have any stray "-l" characters or anything that doesn't begin with "-l" or "-L", so if you're getting additional errors, check the LDLIBS line for any of those. We've found that the configure script sometimes tries to link to a library as -lrt" (yes, it has a quotation mark hanging off the back) when using the Intel compilers and this will usually cause the installation to fail. If you see one or more flags in your LDLIBS line that includes a single quotation mark, just delete the entire flag.

Chapter 4

Conceptual Model

This section is largely inspired by Kollet and Maxwell (2006)

4.1 Water balance

Largly exept for this section which come from Maxwell (2010)

$$\Delta S_{surface}+\Delta S_{subsurface} = Q_{precip}-Q_{runoff}+Q_{run-on}-ET-Q_{recharge}-Q_{S-out}+Q_{S-out} \tag{4.1}$$

$$\Delta S_{surface} + \Delta S_{subsurface} = Q_{precip} - Q_{runoff} - ET \tag{4.2}$$

4.1.1 Richards' equation

Need to have a look at : Jones and Woodward 2001 and Groundwater equation (steady-state, e.g. Ashby and Falgout 1996)

ParFlow use the 3D partial differential equation of Richards' equation. This equation enable to treat saturated and unsaturated within the same equations (i.e.: Variably saturated groundwater flow).

$$S_w(p)S_S\frac{\partial p}{\partial t} + \phi\frac{\partial S_w(p)}{\partial t} - \nabla \cdot \left\{k_r\vec{k_s}\nabla\left(p-z\right)\right\} = q_s \tag{4.3}$$

where:

- p is the pressure head $[L]$,
- z is the elevation $[L]$,
- S_w the water saturation , degree of saturation or relative saturation$[-]$ see Section 6.6. It is usually represented by the Van Genuchten relationship (although ParFlow can use other relations)

$$S_w(p) = S_{res} + \frac{S_{sat} - S_{res}}{\left(1 + (\alpha p)^n\right)^{1-1/n}} \tag{4.4}$$

  - S_{sat} relative saturated water content $[-]$,
  - S_{res} residual saturated water content $[-]$,
  - α is a soil parameter $[L^{-1}]$
  - n is another soil parameter $[-]$

- S_S specific storage coefficient $[L^{-1}]$,
- ϕ the porosity$[-]$,
- $\vec{k_s}$ intrasect permeability tensor $[-]$,
- k_r is the relative permeability $[LT^{-1}]$,

$$k_r(p) = \frac{\left(1 + \frac{(\alpha p)^{n-1}}{(1+(\alpha p)^n)^{1-1/n}}\right)^2}{(1+(\alpha p)^n)^{\frac{1-1/n}{2}}} \tag{4.5}$$

- q_s is the water source/sink term $[T^{-1}]$

In ParFlow, and ParFlow uniquely, the physical quantity does not have a specific unit.So, be careful to remain consistent. For more information we coupled to CLM see section 6.8 .

Both Dirichlet (first-type: pressure specified) and Neumann (second-type: flow specified) boundary condition type can be used on all boundaries. Yet, for the ground boundary condition, Neumann is used with:

$$q_{BC} = \vec{K}(p)(\nabla p - z) \tag{4.6}$$

4.1.2 Overland Flow

Overland is fully coupled with an flow boundary condition. This is explain in subsection 4.1.2

Shallow overland flow is represented by the 2D-kinematic wave equation. Hydrostatic condition $(y = p)$ is assumed and therfore the continuity equation is given by:

$$\frac{\partial p_s}{\partial t} = \nabla \cdot (\vec{v} p_s) + q_r + q_e \tag{4.7}$$

- p_s is the pressure head at surface $[L]$,
- $\vec{v}$ is the velocity at surface $[LT^{-1}]$. Using the kinematics wave approximation $(S_f = S_0)$, Velocity can be calculated with Manning equation (cite)

$$v_{x,y} = \frac{\sqrt{S_{f,(x,y)}}}{n} p_s^{2/3} \tag{4.8}$$

  - n is the Mannings coefficient $[TL^{-2/3}]$
  - S_f is the friction slop $[-]$

- q_r is the source/sink term (rainfall, ET...) $[LT^{-1}]$
- q_r is exchange rate with subsurface $[LT^{-1}]$

4.1.3 Exchange flux

Kollet (2006) introduced a general formulation to fully coupled the surface-subsurface system. This formulation eliminate numerical problem of the conductance type approach.

A continuity in pressure ($p = p_s$) and flux ($q_{bc} = q_e$) between Richard's and the Kinematic wave is set up and the boundary condition is change. The overland equation can be re-written to gives the flux as:

$$q_e = \frac{\partial \|p, 0\|}{\partial t} - \nabla \cdot (\vec{v}\|p, 0\|) - q_r \tag{4.9}$$

Which using the Neuman boundary condition gives:

$$-\boldsymbol{K}(p)\nabla(p - z) = \frac{\partial \|p_s, 0\|}{\partial t} - \nabla \cdot (\vec{v}\|p_s, 0\|) - q_r \tag{4.10}$$

The $\|A, B\|$ operator choose the greater value between A and B.

This formulation create the effect that the last cells is at the same time soil pressure and overland flow pressure

4.2 Energy Balance

This section is inspired by Kollet (2008) which explain how CLM is coupled to ParFlow.

$$R_n = H + L_v E_t + G \tag{4.11}$$

- R_n is the net radiation $[W/m^2]$

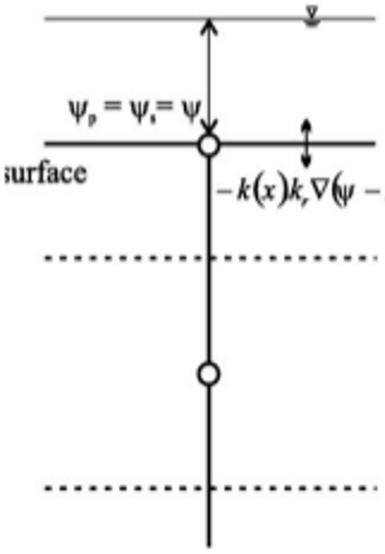
$$R_n = S + L^{\downarrow} - L^{\uparrow} \tag{4.12}$$

  - S is the solar radiation $[W/m^2]$
  - $L^{\downarrow}$ is the incoming Long-wave radiation $[W/m^2]$
  - $L^{\uparrow}$ is the outgoing Long-wave radiation $[W/m^2]$

- H is the sensible heat $[W/m^2]$

$$H = H_c + H_g \tag{4.13}$$

Figure 4.1: from Kollet (2006)



  - H_c is sensible heat flux from vegetation $[W/m^2]$
  - H_g is sensible heat flux from ground $[W/m^2]$

- L_v is the latent heat of evaporation
- E_t is total evaporation $[W/m^2]$

$$E_t = E_c + E_g = E_w + E_{tr} + E_g \tag{4.14}$$

  - $E_c = E_w + E_{tr}$ is the evaporation from vegetation
  - E_g is the evaporation from the ground
  - E_w is the evaporation from wet foliage
  - E_{tr} is the transpiration

- G is the ground heat flux $[W/m^2]$, with λ soi thermal conductivity and T_{grnd} ground temperature

$$G = \lambda T_{grnd} \tag{4.15}$$

Mahfouf and Noilhan 1991 (alpha beta method)

4.3 Surface Model: CLM

Two fully coupled the water balence of equation ... and the energy balence of equation... the sink/source term relete with total evaportation E_t and infiltration q_g:

$$q_s = L_v E_t + q_g \tag{4.16}$$

All the energy term (R_n, H, G) are only depending on the water content $(\theta or S_w)$ and with Van Genuchten at the pressure p which is share by PF to CLM.

Yet, few assumptions/simplifications are done:

- Independance of hydraulic conductivity k_r and temperature T.
- neglect the convective component in the ground heat flux
- neglecting explicit vapor transport in ground evaporation.

Chapter 5

Numerical Implementation

I will try to explain or at least gather the information to describe the numerical method used.

Parflow is using *"...an implicit backward Euler and cell-centered finite difference scheme for the discretization in time and space respectively. The equations are solve using a Newton-Krylov method with multigrid preconditioning"(Kollet, 2008).*

5.1 Numerical method to solve PDE

The general expression of an partially differential equations (PDE) is given as follow where the highest derivated degree is the order of the PDE (first: $\frac{\partial u}{\partial x_1}$ and second: $\frac{\partial^2 u}{\partial x_1 \partial x_n}$)

$$F\left(x_1, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_n}, \ldots\right) = 0 \qquad (5.1)$$

The Finite methods are numerical technique for finding approximate solutions of PDEs.

5.1.1 Finite Difference Method (FDM)

Finite Difference method (FDM) uses finite difference equations to approximate derivatives. see section 5.1.4

5.1.2 Finite Volume Method (FVM)

Finite Volume method (FVM) uses the Divergence theorem at each node considered as a volume. FDM is a special case of FVM.

5.1.3 Finite Element Method (FEM)

Finite Element method (FEM) is transforming the PDE into a system of simple function. It uses variational methods (the calculus of variations) to minimize an error function and produce a stable solution.

5.1.4 Numerical method to solve ODE

The general expression of an ordinary differential equations (ODE) is given as follow where n is the order

$$F(x, y, y', y'', ...y^{(n)}) = 0 \tag{5.2}$$

- an the Euler Method is a numerical procedure to solve ODE of the first order ($n = 1$) given an initial condition with:

  $$y_{n+1} = y_n + (x_{n+1} - x_n) f(x_n, y_n) \tag{5.3}$$

- Implicit vs explicit: Explicit methods calculate the state of a system at the next time from the state of the system at the current time (ie: $y_{n+1} = f(y_n)$), while implicit methods find a solution by solving an equation involving both the current state of the system and the later one (ie: $g(y_{n+1}, y_n) = 0$).

- Backward vs forward: are form of the finite difference expression. They differ in the choice of the node choose to solve the derivate. forward, central and backward:

  $$\left.\frac{dy}{dx}\right|_n = \frac{y_{n+1} - y_n}{\Delta x} \tag{5.4a}$$

  $$\left.\frac{dy}{dx}\right|_n = \frac{y_{n+1} - 2y_n + y_{n-1}}{2\Delta x} \tag{5.4b}$$

  $$\left.\frac{dy}{dx}\right|_n = \frac{y_n - y_{n-1}}{\Delta x} \tag{5.4c}$$

  As you must have notice, backward is a type of explicit and forward a type of implicit method. (they are not the same thing !)

- The implicit backward Euler method computes the approximation:

  $$y_{n+1} = y_n + (x_{n+1} - t_n) f(x_{n+1}, y_{n+1}) \tag{5.5}$$

  As you can see in this equation the term y_{n+1} appear in both side which make it implicit.

- Space and time: In the case where there is two independent variables (x_1, x_2 or x, t), the equation to solve is a partial differential equation (PDE) ($F(x, t, y, \frac{dy}{dx}, \frac{dy}{dt}) = 0$) . The approximation can therefore be different for space and time. In our case we have:

$$\left.\frac{dy}{dt}\right|_n = \frac{y_n - y_{n-1}}{\Delta t} \tag{5.6a}$$

$$\left.\frac{dy}{dx}\right|_n = \frac{y_{n+1} - 2y_n + y_{n-1}}{2\Delta x} \tag{5.6b}$$

$$\tag{5.6c}$$

5.2 Newton and Kyrlov method

The Newton method is used to solve the equation (Euler method give an approximate equation to solve, Newton solves it !). In the general term, solving mean find u such as $F(u) = 0$. (the u term would be in our case y_{n+1}) A description of the Newton method: *"one starts with an initial guess, then the function is approximated by its tangent line (derivate), and one computes the x-intercept of this tangent line. This x-intercept will typically be a better approximation to the function, and the method can be iterated."* In another word :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{5.7}$$

If we want to go one step futher, we need some other tools such as the Jacobian matrix which gather all the different derivate of an vector function $F = [F_1(x_1, ..., x_n), ..., F_m(x_1, ..., x_n)]$ in the following way:

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}. \tag{5.8}$$

Using this the Newton method can be upgrade for vector function which appear to be a linear system ($Ax = b$ where $A = J(x)$, $dx = x$ and $-F(x) = b$)

$$J(x)dx = -F(x) \tag{5.9}$$

In order to find the Jacobian matrix, numerical approximation are often required (such as finite difference seen before). Yet this computations involve a lot of computation and memory and therefore approximation of the Jacobian is often used.

Krylov method is solving a linear system ($Ax = b$) in an iterative way which requires only matrixvector products (not individual element) to carry out the iteration. In 2004, Knoll and Keyes(ref:) reviewed the Jacobian-free Newton-Krylov method (JFNK) which is used in Parflow. I can't go in more detail because I don't fully understand. Google it if you want to know more.

5.3 Preconditioning

Th last notion that we are going to talk about is the preconditioning. It is a procedure to condition a given problem into a form that is more tractable

Multigrid methods accelerate the iterative solution by solving a coarse version of the problen and using using that to inform the fine grid.

could be a Harmonic weighting and a one point-upstrean weighting of the relative permeabilites (Kellet,2006, p949). finite difference/finite volume 3D flow

Jones and Woodward (2001)

5.4 Parralelism

: (Falgout and Jones (1999)) ashby and Falgout 1996 the numerical algorithm (solver) must be scale linearly with problem size Message passing distributed memmory using a Octree data structure

Conclustion: speed up : generally 10x faster than 2D and 100x for 3D

Chapter 6

Revised Guide of a PARFLOW script

In this chapter, I will go through all the step to create the different input file.

6.1 Scheme of the grid

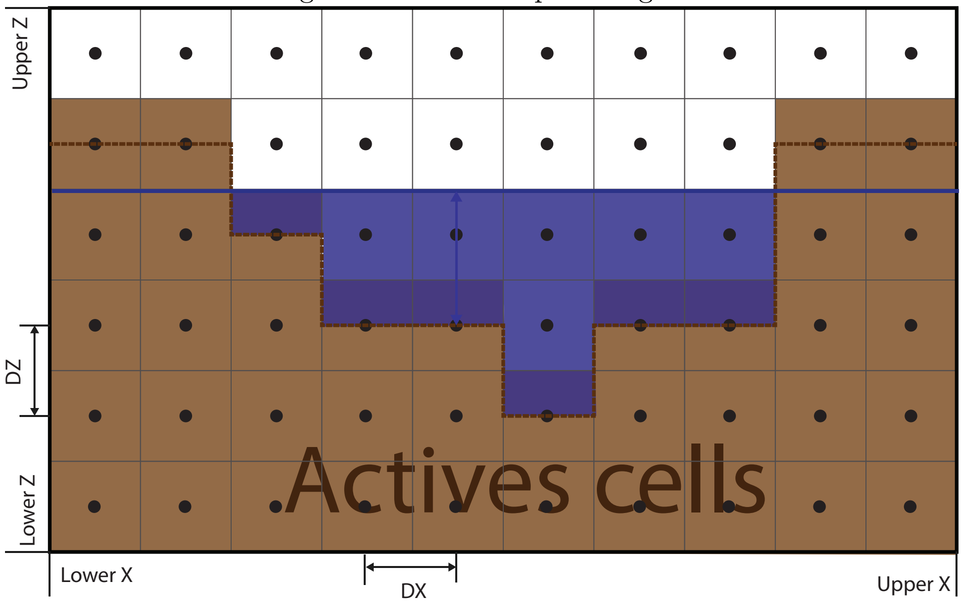
The Figure 6.1 displays a sample grid illustrating all the grid information. I would like to draw you attention to the following points:

- It is a cell-center scheme. ($x_1 = dx/2$)
- The land surface stop at the center of the cell while the water table (when p¿0)is between two cell.
- the overland flow occur in half of the last cell. But all the cell above are inactive(eventhough their is pond-water in them)
- CLM run over the ten first cell whatever grid resolution there is.

6.2 Summury of Input/Ouput

Table of input output

Figure 6.1: 2D exemple of a grid



Key	Symbole	Description	Unit	type
		Grid Information		
Lower X,Y,Z		position of the origin		double
NX,NY,NZ		number of grid cells		integer
DX,DY,DZ		size of a grid cells	$[L]$	double
		Surface Information		
shape		mesh of the surface shape built with `...`)	.pfsol	
slope x, y		value of the slope can be build with `pfslopex` by PF from DEM (3D not 2D)		2D-pfb or double
manning	n	value of the manning coefficient	$[-]$	2D-pfb or double
		Soil information		
perm	k_r	saturated hydraulic conductivity	$[L/T]$	3D-pfb or double
porosity	ϕ	porosity of the soil	[-]	double
alpha, n	α, n	parameters for Van Genuchten if 3D, needs to be on domain	[?]	double or 3D-pfb
SRes, SSat	S_{res}, S_{sat}	Saturated water content (see subsection 6.6)	[-]	double or 3D-pfb
		Met forcing		
DSWR	$L^{\downarrow}$	Long wave radiation incoming	$[W/m^2]$	1D-t
DLWR	$K^{\downarrow}$	Short wave radiation incoming	$[W/m^2]$	1D-t
APCP	P	Precipitation	$[mm/s]$	1D-t
Temp	T_a	Air temperature	$[K]$	1D-t
UGRD		N-S wind speed	$[m/s]$	1D-t
VGRD		E-W wind speed	$[m/s]$	1D-t
Press		Atmospheric Pressure	$[Pa]$	1D-t
SPFH		Specific humidity	$[kg/kg]$	1D-t

function	Key	Symbole	Description	Unit	type
			Grid Information		
`WriteSiloMask`	mask		matrix of 1 (soil) and 0 (air)		3D-0T
`pfcomputetop mask`	top		matrix of the top soil elevation		2D-0T
`pfextracttop 3Dfield`	top-layer		extract the ground value for any 3D field (can be loop over time)		2D-0T
			CLM output		
`WriteSiloCLM`	`eflx_lh_tot`	Lv	latent heat flux total	$[W/m^2]$	3D-0NT
`WriteSiloCLM`	`eflx_lwrad_out`	$L^{\uparrow}$	outgoing long-wave radiation	$[W/m^2]$	3D-NT
`WriteSiloCLM`	`eflx_sh_tot`	H	sensible heat flux total	$[W/m^2]$	3D-0NT
`WriteSiloCLM`	`eflx_soil_grnd`	H_g	ground heat flux	$[W/m^2]$	3D-0NT
`WriteSiloCLM`	`qflx_evap_tot`	L_vE_t	total evaporation	$[W/m^2]$	3D-0NT
`WriteSiloCLM`	`qflx_evap_grnd`	L_vE_g	ground evaporation without sublimation	$[mm/s]$	3D-0NT
`WriteSiloCLM`	`qflx_evap_soi`		soil evaporation	$[mm/s]$	3D-0NT
`WriteSiloCLM`	`qflx_evap_veg`	L_vE_c	vegetation evaporation	$[mm/s]$	3D-0NT
`WriteSiloCLM`	`qflx_tran_veg`	$E_{tr}L_v$	vegetation transpiration	$[mm/s]$	3D-0NT
`WriteSiloCLM`	`qflx_infl`	q_g	soil infiltration	$[mm/s]$	3D-0NT
`WriteSiloCLM`	`swe_out`	S	snow water equivalent	$[mm]$	3D-0NT
`WriteSiloCLM`	`t_grnd`	T_g	ground surface temperature	$[K]$	3D-0NT
`WriteSiloCLM`	`t_soil`	T_s	lsoil temperature over all layers	$[K]$	3D-0NT
			PF ouput		
`WriteSiloSubsurfData`	pressure	$Press$	pressure	$[L]$	3D-NT
`WriteSiloSubsurfData`	saturation	S	saturation ($S_{res} < S < S_{sat}$)	$[-]$	3D-NT
`pfhhead pressure`	head	H	head (P+z)	$[L]$	3D-NT
`pfflux perm head`	flux	q	Darcy flow	$[L/T]$	3D-NT
			PF computed		
`pfsubsurfacestorage ...`	surface runoff	R	runoff out of domain	$[L^3/T]$	3D-NT
`pfsurfacestorage ...`	surface storage	SS	surface water storage (ponding)	$[L^3]$	3D-NT
`pfsurfacerunoff ...`	subsurface storage	SSS	sub-surface water storage	$[L^3]$	3D-NT
`pfsum ...`			sum of any 2D or 3D field	$[L^3]$	0D

6.3 Architecture of files

In an very unstructure manner, PARFLOW find the input and output on folder (where the main script is). But like me I encourage you to structure a little bit more your data.

- `Pre/` is the folder containing the preparation step for the data
- `Required_data/`: is the folder with external data (DEM, met forcing)
- `pre.m`: is the matlab script to create the input data.
- `generate_function/`: folder with all the matlab function which generate data
  - `generate_dem()`:
  - `generate_soil()`:
  - `generate_met()`:
  - `generate_dem()`:
- `Input/`: is the folder with all the file required for the TCL script produced by my preparations steps. Parflow will then just copy these file into in main folder
- `Output/`: is the folder where the run will create all the new data
  - `TXT`:
  - `CLM output`:
- `Post/`:
  - `load_data()`:
  - `open_SA()`:
  - `plot`:

6.4 Various notes and remarques

- I was not able to make this expression work for decimal value (even with transforming into double)

  ```
  set DY  [expr ($UpperY - $LowerY) / $NY]
  ```

- For all the .SA file imported it is mandatory to set the grid for them like this

  ```
  pfsetgrid {nx ny nz} {x0 y0 z0} {dx dy nz} {x0 y0 z0} {dx dy dz} $dem
  ```

- There is a nice little option with the solver:

  ```
  Pfset Solver.TerrainFollowingGrid                True
  ```

-

6.5 CLM files

Fours files are necessary for CLM:

- `drv_vegp.dat`: vegetation parameter data file
- `drv clmin startup dat`: CLM parameters see CLM manual :Appendix 4: IGBP Land Cover Types Definition. ParFlow only allows one land cover type per grid cell: there should be one column per row with a value of 1.0, and all other values of 0.0. Sand and clay were previously used for the hydrology part of CLM, but have been replaced by ParFlow, and so dummy values are used.
- `drv_vegm.dat`: specify vegetation type
- `narr_1hr.tx` tMeteorological Forcings ()
- Sand and clay were previously used for the hydrology part of CLM, but have been replaced by ParFlow, and so dummy values are used. (about `drv_vegm.dat`)
- There should be one column per row with a value of 1.0, and all other values of 0.0. This is because ParFlow only allows one land cover type per grid cell.
- The ending time can be set far in the future, because the simulation ending time is controlled by ParFlow (`drv_clmin.dat`)
- For the CLM files, manual distribution of the file need to be done

  ```
  set num_processors [expr [pfget Process.Topology.P] * [pfget Process.
  for {set i 0} { $i <= $num_processors } {incr i} {
          file delete drv_clmin.dat.$i
          file copy drv_clmin.dat drv_clmin.dat.$i
  }
  ```

Name	Symbol	Definition	Equation
Porosity	$\phi[\%]$	void over total	$\eta = \frac{V_v}{V_t}$
Water content	$\theta[\%]$	water over total	$\theta = \frac{V_w}{V_t}$
Water saturation	$S_w[\%]$	water over void	$S_w = \frac{V_w}{V_v} = \frac{\theta}{\eta}$
Normalized water content or effective saturation	Θ or $S_e[\%]$		$\Theta = \frac{\theta - \theta_{res}}{\theta_{sat} - \theta_{res}}$

6.6 Deal with Sres and Ssres

In theory, θ_{res} is the residual water content, defined as the water content for which the gradient $d\theta/dh$ becomes zero; and, θ_{sat} is the saturated water content, which is equivalent to porosity, ϕ. In practice, we fit the Van Genuchten curve to data and find these parameters.

As a note, the `pfwatertabledepth` function of ParFlow computes, as explain in the manual the distance from the top to first cell with saturation=1. First, this isn't right in my opinion as the water-table should be computed from the first cell where pressure=0, and secondly, when you put a saturated water content less than 1, this function is not working. (So don't use it)

6.7 Deal with Specific humidity

The mass of air (m_a) is the sum of a mass of vapor (m_v) and all the other molecules (mainly N_2 and O_2) that form the dry mass (m_d).

$$m_a = m_v + m_d \tag{6.1}$$

- The *mixing ratio* is the ratio of vapor mass over dry mass.

$$\mathrm{MR} = \frac{m_v}{m_d} \tag{6.2}$$

- The definition of *Specific humidity* is the ratio of mass of vapor over mass of air (value around 0.01).

$$\mathrm{SH} = \frac{m_v}{m_a} \tag{6.3}$$

- More often the *relative humidity* is often used. In this case humidity is the ratio of mixing ratio of the air over the mixing ratio of the same air brought to saturation of water. With is the same as the ratio of specific humidity over saturated specific humidity

$$\mathrm{RH} = \frac{P_v}{P_v^{sat}} = \frac{\mathrm{MR}}{\mathrm{MR}^{sat}} \tag{6.4}$$

How to convert between them ? First, as the mass of vapor is so small compare to the mass of air, it is commonly assumed that:

$$\text{SH} = \frac{m_v}{m_d + m_w} = \frac{\text{MR}}{1+\text{MR}} \approx \text{MR} \tag{6.5}$$

From Dalton's law, (total) pressure is equal to the sum of partial pressure ($P_a = P_v + P_d$) where each partial pressure is proportional to the his molar ratio (with $M_d = 0.028964$ and $M_w = 0.018016$ are the molecular weight $[kg/mol]$):

$$\frac{P_v}{P_a} = \frac{n_v}{n_a} = \frac{m_v/M_v}{m_v/M_v + m_d/M_d} = \frac{\text{MR}}{\text{MR} + M_v/M_d} \tag{6.6}$$

Therefore the mixing ratio can be calculated with air (or total) pressure and vapour pressure. This can be futher simply considering $P_a \gg P_v$

$$\text{SH} \approx \text{MR} = \frac{M_v/M_d}{P_a/P_v - 1} = \frac{M_v}{M_d}\frac{P_v}{P_a} \tag{6.7}$$

Sometime vapour pressure is not known but relative humidity is. In this case the saturated vapour pressure can be computed with empirical relationship which required temperature $[C]$ and pressure $[Pa]$ for better accuracy. Here is Buck's equation:

$$P_v^{sat} = (1.0007 + 3.46 \times 10^{-4} P_a) \times (6.1121) e^{\left(\frac{17.502T}{240.97+T}\right)} \tag{6.8}$$

$$\text{SH} \approx \text{MR} = \text{RH} \times \text{MR}^{sat} = \text{RH}\frac{M_v}{M_d}\frac{P_v^{sat}}{P_a} \tag{6.9}$$

6.8 Deal with time

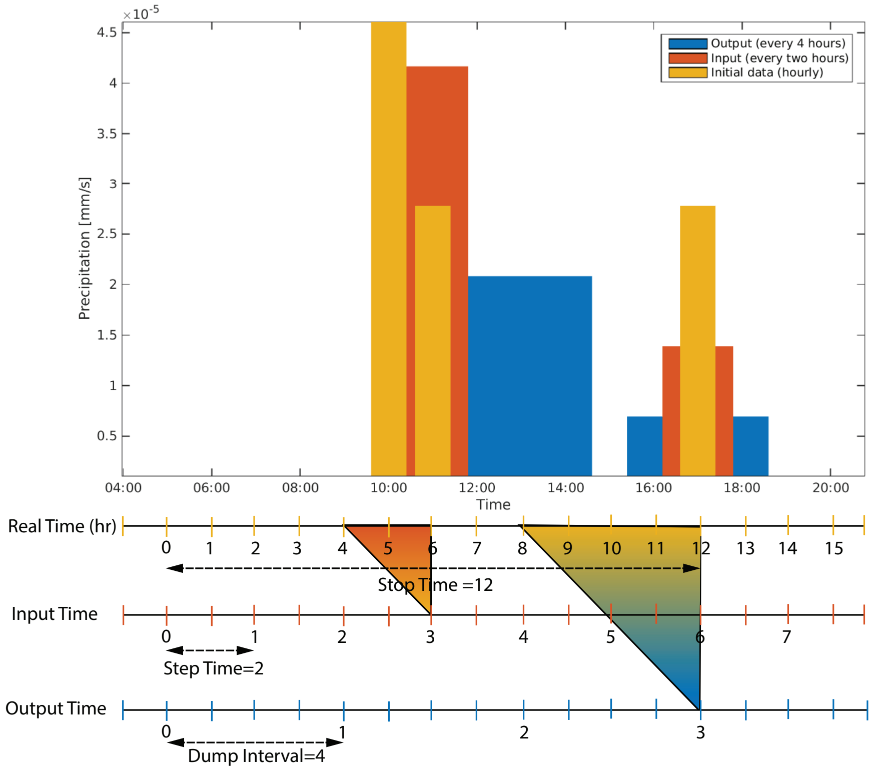
First read the manual to understand everything about timing info and time cycle (section 5.1.3). I will try to explain what i understood so far: ParFlow, on its own, can work in any unit of length and time as long as your are consistent. Yet, CLM input and output have specific unit (m,hr,mm/s, W/m^2...). And the default time is [HOUR] and space is [M]. Therefore all informations of time are given in unit of hour, even in ParFlow.

- The *Stop Time* is the length (in hour) of the simulation.
- The *Step Time* gives indication at when PF will compute ; this information will be given to CLM also (see outputfile `washita.para.out.dat.0`). CLM also run an internal clock where you need to provide the exact day of the year in `drv_clmin.dat` (The ending time can be set far in the future, because the simulation ending time is controlled by ParFlow).

- The *Dump Interval* inform PF at when to output files. If the dump interval is less than the Step time, parflow will output the same result. It is also possible to set step Time to less than one, 0.5 for example to run every half hour (but then you need half-hourly input data)

In the example of Figure 6.2, PF compute every two hour a run from input given at every two hour (In my case I averaged precipitation over two hours). The output are being produced every 4 hours, and therefore I averaged precipitation over the 4 hours in order to compare it to ouput (eg. Infiltration or ET).

Figure 6.2: Illustration Time



6.9 Surface file

Parflow needs 3 files related to the surface. You can fin python and fortran code helping you with that with John Koudelka or in an exemple of ParFlow

- Slope X and Y.
  - I think that I have understand that as the overlandflow is computed with the kinematic wave equation, flat zone or pit will arise

in a large increase of water in this cell (peak). Parflow has a command to remove any flat ot local minima elevation. (see manual for more info). (be carfull, it needs to be a 4 cells not 8 cells as usual pit removal tools)

```
set flatfill [pffillflats $dem]
set pitfill [pfpitfilldem $dem 0.01 50]
```

- The solid file (.pfsol) is similar to a Triangulated Irregular Network (TIN) but includes subsurface layers beneath the dataset.