# ParFlow System Basics

*How to setup and run a model and where to get help*

ParFlow Short Course

Module 2

# Learning Objectives:

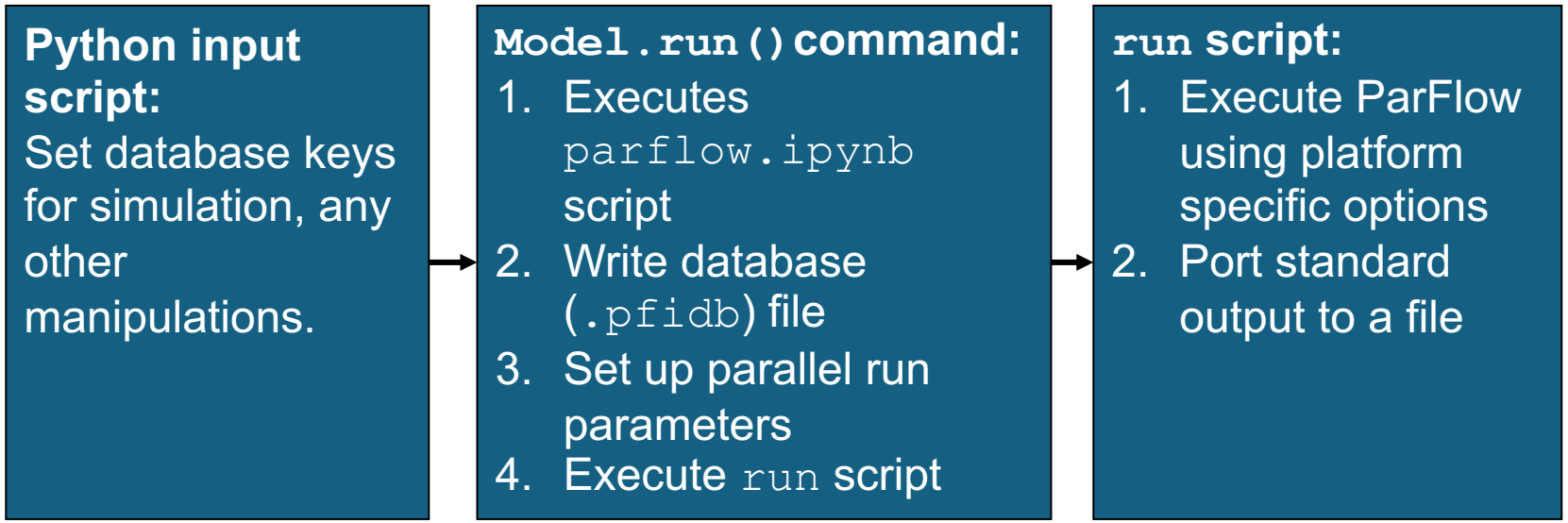
At the end of this module students will understand:

- That there are a suite of options for working with ParFlow models
- ParFlow models can be setup and run using TCL or Python (Or directly from a PFIDB)
- The difference between setting keys and running a model
- Options for installing and running ParFlow on multiple systems
- What PFTools are and how you can download and access them
- What the PFTools python package is and how to access it
- Where to get help on ParFlow Keys and PFTools
- Where to find example parflow scripts in tcl and python

# Module Outline:

1. *Setting up a model input file*
2. *Running a model*
3. *What comes out and how to look at it*
4. *Tools for working with ParFlow models*
5. *Online resources*

# 1. Setting up a model input file *(How you tell ParFlow what to do)*



# Input Scripts

- TCL/TK scripting language with python interface
- All parameters input as keys using `pfset` command (tcl) or `model.”key_name”` (python)
- Keys used to build a database that ParFlow uses
- ParFlow executed by `pfrun` command (for tcl) and `model.run()` for python
- Since input file is a script may be run like a program

# Example: Setting up the input grid

```
#---------------------------------------------------------
# Computational Grid
#---------------------------------------------------------
#Locate the origin in the domain.
model.ComputationalGrid.Lower.X = 0.0
model.ComputationalGrid.Lower.Y = 0.0
model.ComputationalGrid.Lower.Z = 0.0

# 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

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

Coordinates (length units)

Cell size (length units)

Grid dimensions (integer)

# Example: Setting up the timing

```
#---------------------------------------------------------
# Setup timing info
#---------------------------------------------------------

model.TimingInfo.BaseUnit = 1.0

model.TimingInfo.StartCount = 0

model.TimingInfo.StartTime = 0
model.TimingInfo.StopTime = 72.0

model.TimingInfo.DumpInterval = 1.0

model.TimeStep.Type = "Constant"

model.TimeStep.Value = 1.0
```

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

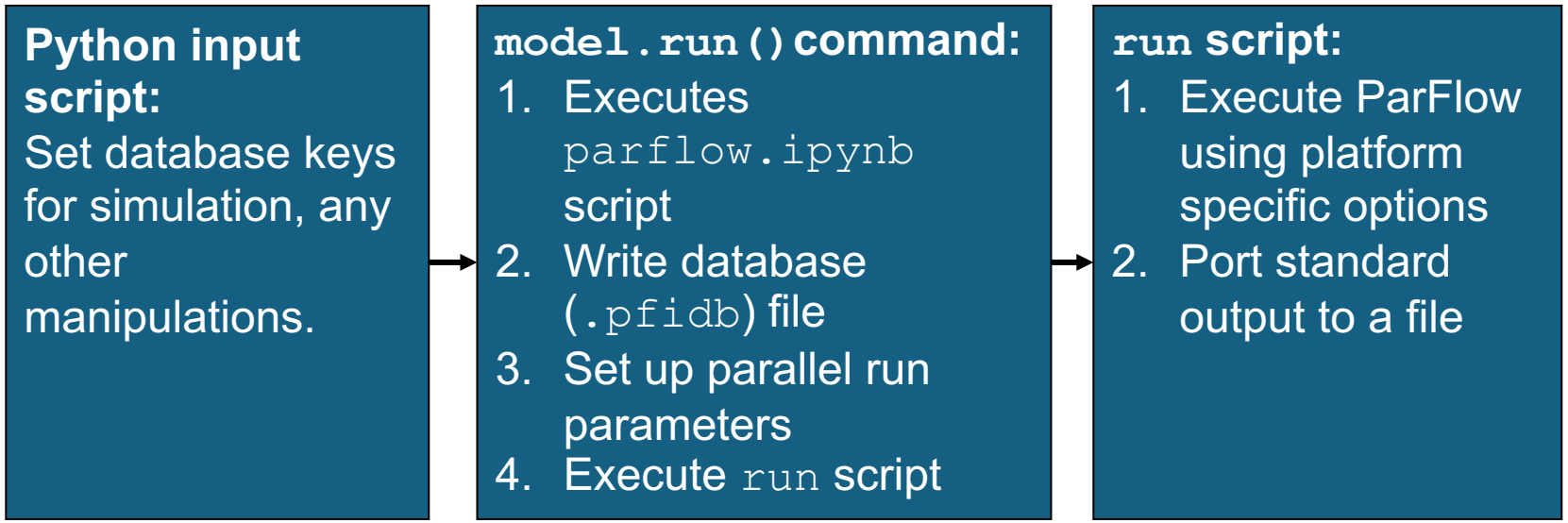
Timestep type

ΔT (T)

# Best practices for building an input file:

- Start from an existing script:
  - Look at the annotated input scripts in the manual
  - Look at the test problems that come with ParFlow (See list in section 3.5)
  - Use scripts for test problems presented in this course
- Get the details on every input key from the manual (Section 6)

# 2. Running ParFlow simulations (*How to press go*)



# Running ParFlow

- `model.run()` 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)

**Mandatory Content at the top of your python script:**

```
# Import required packages

import os
import numpy as np
from parflow import Run
import shutil from parflow.tools.fs
import mkdir, cp, get_absolute_path, exists from
parflow.tools.settings
import set_working_directory
…
model.run()
```

Load the recommended packages to run parflow in python

Run parflow, execute run command on model object

**To run the model:**

model.run()

# 3. Handling outputs
# (*What comes out and how to look at it*)

# 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
```

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

Perm/porosity files:

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

Other/diagnostic files:

```
myrun.pfidb
myrun.out.pftcl
myrun.out.txt
```

Parflow database

Line output

# Running ParFlow (file structure NetCDF)

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

For a project called ‘myrun’ the log and database files are the same

Initial Pressure+Saturation+Subsurface files:

myrun.out.00000.nc

Output time step, 00000 is initial, and contains static output (like perm) for NETCDF

```
myrun.out.00000 {
dimensions:
        time = UNLIMITED ; // (1 currently)
        x = 1 ;
        y = 1 ;
        z = 20 ;
variables:
        double time(time) ;
        double pressure(time, z, y, x) ;
        double saturation(time, z, y, x) ;
        double perm_x(time, z, y, x) ;
        double perm_y(time, z, y, x) ;
        double perm_z(time, z, y, x) ;
        double porosity(time, z, y, x) ;
        double specific_storage(time, z, y, x)
;
}
```

Time varying Pressure+Saturation outputs:

myrun.out.press.00001.nc

myrun.out.press.00241.nc

Multiple timesteps per file

```
myrun.out.00241 {
dimensions:
        time = UNLIMITED ; // (240
currently)
        x = 1 ;
        y = 1 ;
        z = 20 ;
variables:
        double time(time) ;
        double pressure(time, z, y, x) ;
        double saturation(time, z, y, x)
;
}
```

# Visualizing Outputs

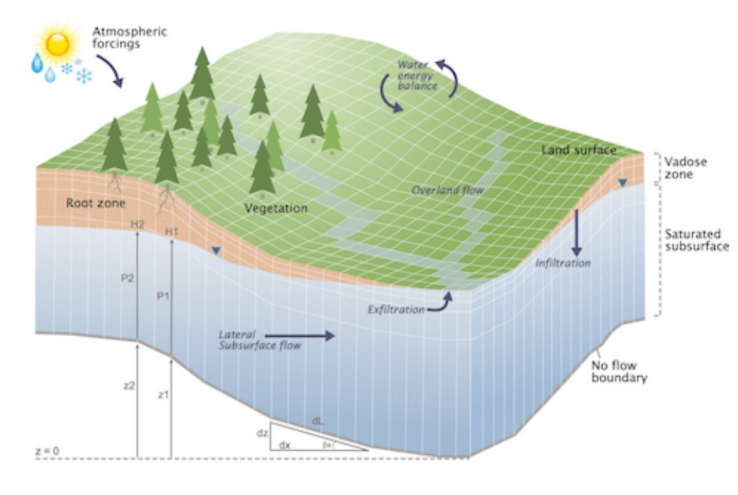
In Python several standard, Pythonic tools are available:

- PFTools, pip-installable, robust toolset to read PFB files
- NETCDF
- XArray

To render files in 3D we recommend ParaView:

- Free, developed by Kitware Inc
- https://paraview.org

# ParFlow Resources




*Conceptual illustration of the ParFlow-CLM model (Figure from Maxwell and Condon (2016) supplementary information)*

## About ParFlow

ParFlow is a numerical model that simulates the hydrologic cycle from the bedrock to the top of the plant canopy. It integrates three-dimensional groundwater flow with overland flow and plant processes using physically-based equations to rigorously simulate fluxes of water and energy in complex real-world systems. You can **learn more about the ParFlow model and how it works here**.

ParFlow is a computationally advanced model that can run on laptops and supercomputers and has been used in hundreds of studies evaluating hydrologic processes from the hillslope to the continental scale.

Website
https://parflow.org

Source Code
https://github.com/parflow/parflow

Read-the-Docs
https://parflow.readthedocs.io/

## Tools for interacting with ParFlow

### PF Tools

A set of tools for pre and post processing ParFlow model results

### SubsetTools

Tools for building watershed models from the National Framework

Linked from:
https://hydroframe.org/parflow-resources

## Short Courses

We teach ParFlow short courses for new users. To be informed on upcoming short courses please join the ParFlow mailing list. In the meantime you can get started with the resources from previous short courses:

### Basic TCL Based

### Advanced TCL Based

### Python Based

## Mailing List

Become a part of our mailing list to stay up to date on the most recent version of ParFlow, and to receive regular updates on resources!
**Send an email to parflow@googlegroups.com with the subject "Subscribe" to join!**