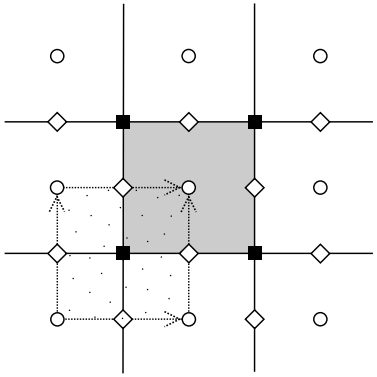
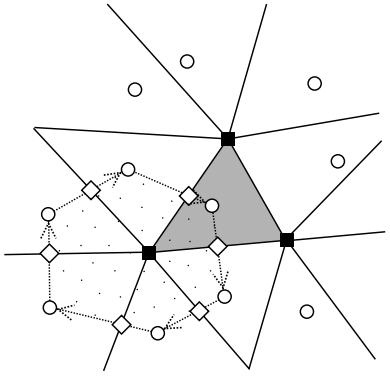


(a) Raster grid

LANDLAB



(d) Delaunay triangle cells with Voronoi polygon patches

A Python library for 2D numerical modeling

Gregory E. Tucker, Daniel E.J. Hobley, Jordan M. Adams,
Nicole M. Gasparini, Eric Hutton,
Erkan Istanbulluoglu, Sai Nudurupati

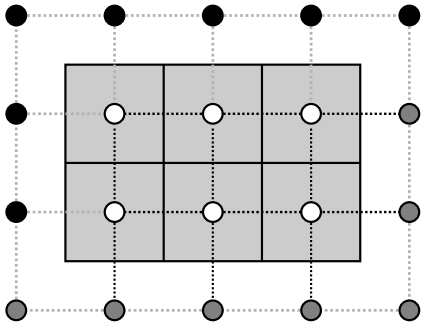
What is Landlab?

- A Python-language programming library
- Supports efficient creation and/or coupling of 2D numerical models
- Geared toward (but not limited to) earth-surface dynamics

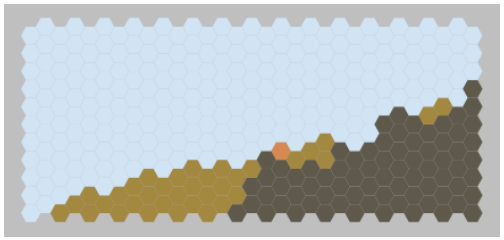
What Landlab provides

1. Grid creation and management
   - Create a structured or unstructured grid in one or a few lines of code
   - Attach data to grid elements
     - Facilitates staggered-grid schemes
     - Passing the grid = passing the data

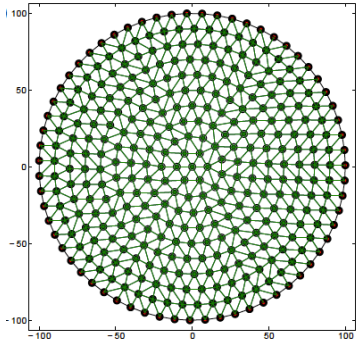
RASTER



HEXAGONAL



RADIAL



What Landlab provides

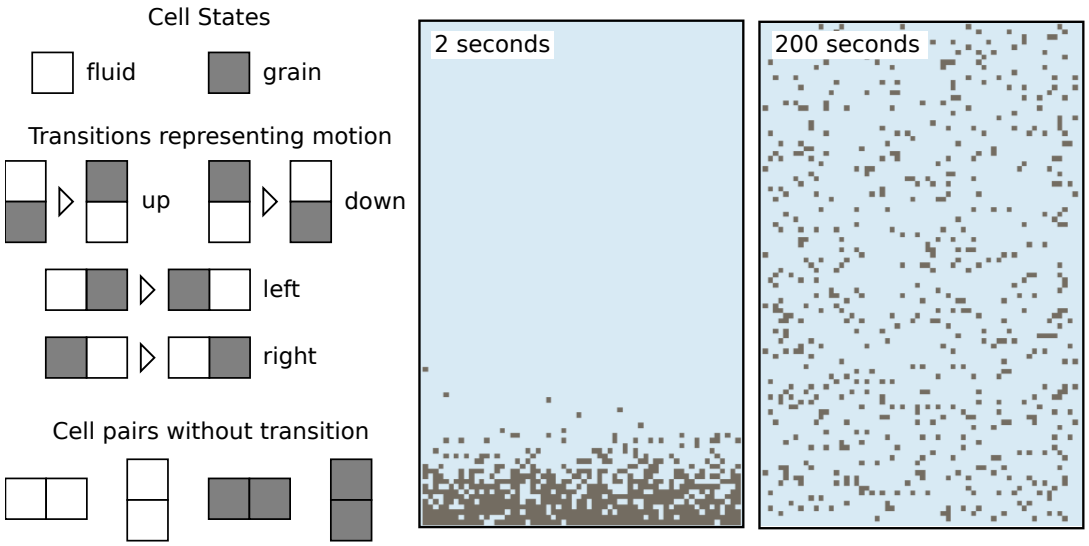
2. Coupling of components
   - A *component* models a single process (e.g., lithosphere flexure, incident solar radiation, flow routing across terrain)
   - Components have a standard interface and can be combined by writing a short Python script
   - Save development time by re-using components written by others

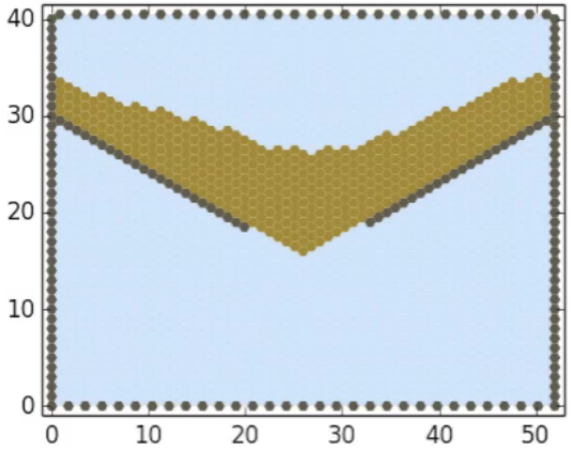
What Landlab provides

3. Input and output
   - Read model parameters from a formatted text file
   - Read in digital elevation model (DEM) ➔ grid
   - Write gridded output to files (netCDF format)
   - Plot data using Matplotlib graphics library

What Landlab provides

4. Support for cellular-automaton modeling
   - CellLab-CTS: Continuous-time stochastic CA model "engine"





What Landlab is **not**

- A model *(it's a library…though the collection includes several components and models)*
- CHILD
- For coupling multi-language, legacy models *(that's WMT)*
- Painfully slow *(Landlab uses Numpy and [soon] Cython for speed)*
- Only for "land" models

Today's clinic

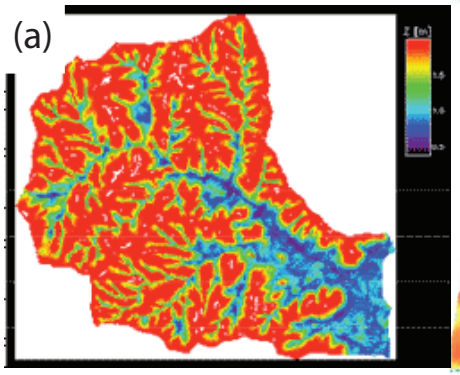
- Introduction and overview of Landlab
  - Background
  - Quick look at Python, Numpy, and Matplotlib
  - Example of Landlab in action: fault scarp diffusion
  - More on Landlab's ModelGrid module
  - Working with Components
- Hands-on exercises

Motivation for Landlab

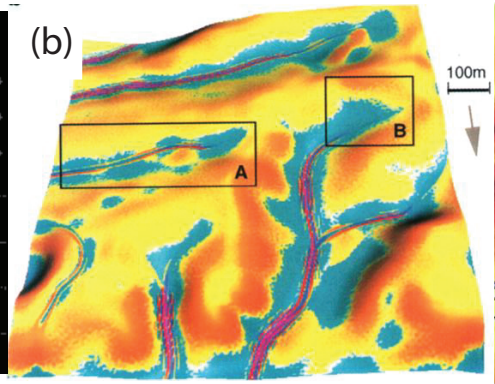
1. The science of Earth-surface dynamics is evolving rapidly—so too must models and their software
2. Software behind numerical models has a cost
3. Earth-surface dynamics models often have common elements across diverse applications

Many commonalities in earth-surface process models

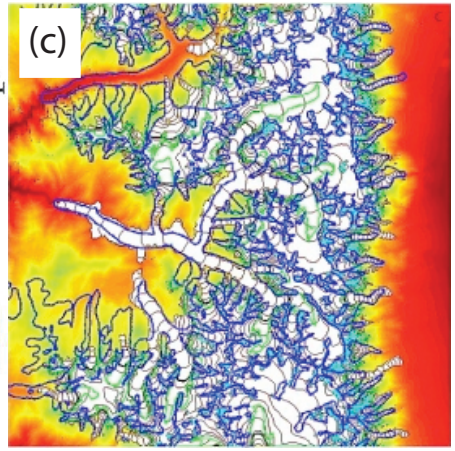
CATCHMENT HYDROLOGY
(Ivanov et al., 2004)

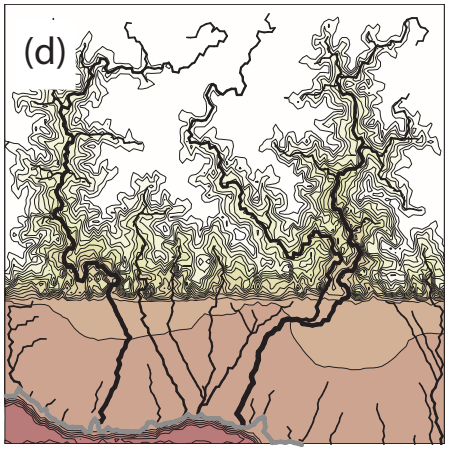


SOIL EROSION
(Mitas and Mitasova, 1998)

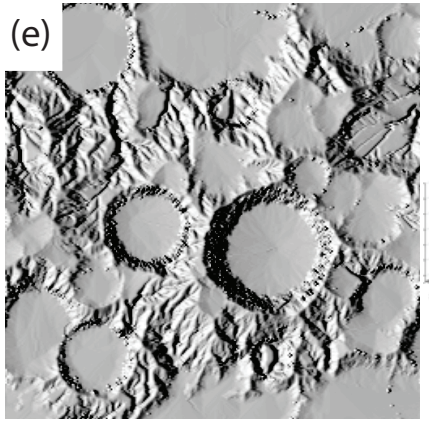


GLACIER DYNAMICS
(Kessler et al., 2006)

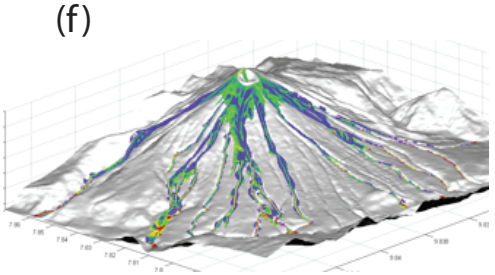




LANDSCAPE EVOLUTION
(Tucker and Hancock, 2010)



IMPACT CRATERING AND DEGRADATION
(Howard, 2007)



LAVA FLOWS
(Kelfoun et al., 2009)

Landlab's solutions

- Use a high-level, open-source language with a rich set of scientific computing libraries
- Package common operations into an open-source library. Examples:
  - Grid creation/configuration, calculating finite-difference gradient on a data field, I/O
- Package useful operations/models into reusable components
- Provide a simple way to combine two or more components that share a common grid and data

To take advantage of these solutions, need to be familiar with:

- Python (including classes)

- Numpy
  (scientific computing library)

- Numerical computing principles

Installing Landlab

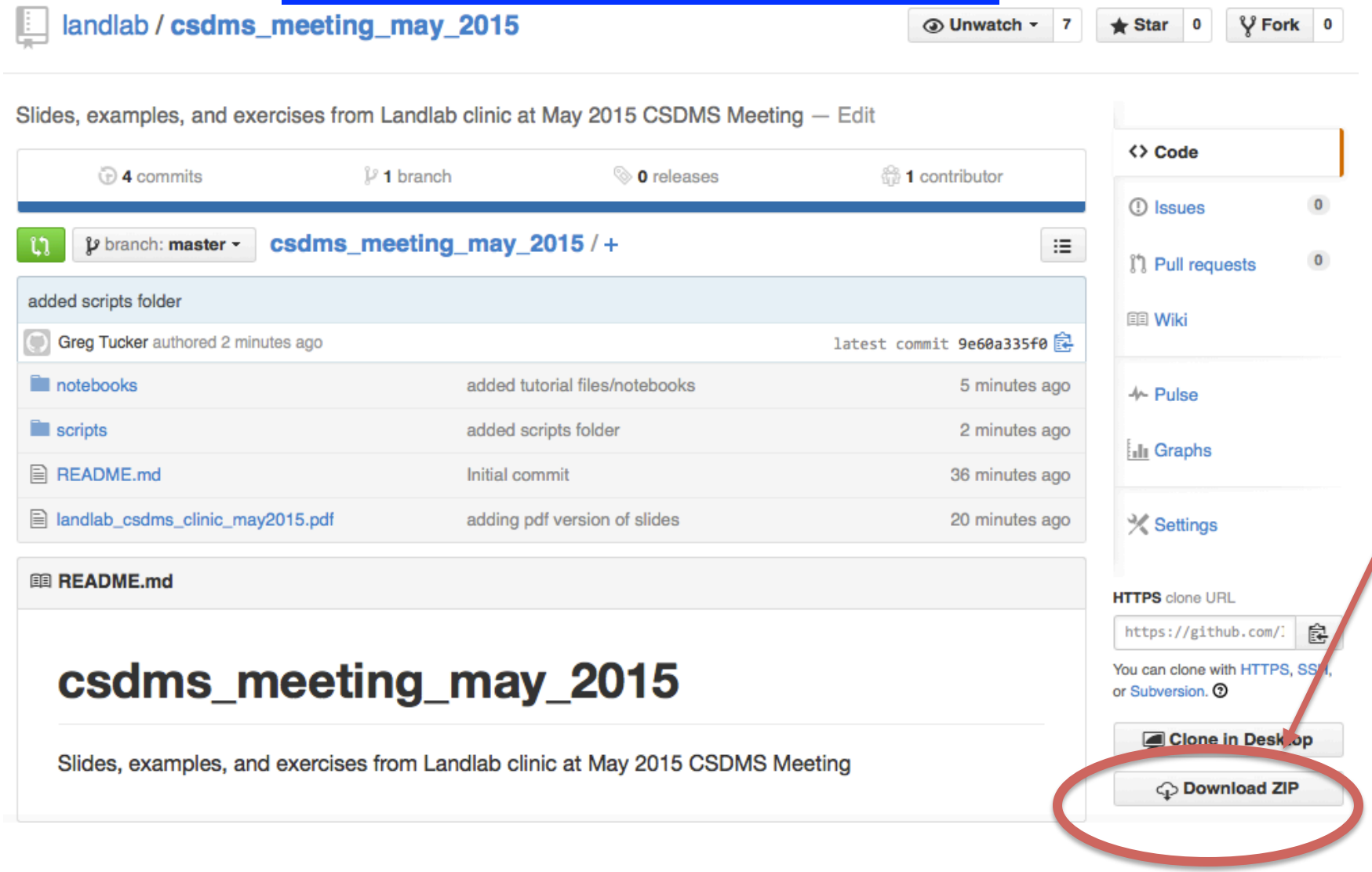
- Installation instructions:
  - http://landlab.readthedocs.org
  - Select **Installing Landlab**
- Basic steps:
  1. Install Python, Numpy, and Scipy
     - easiest method: install Anaconda: http://continuum.io/downloads
     - or Canopy:
     - https://store.enthought.com/

Installing Landlab

- Basic steps:
  2. Verify default Python version
     - Open a terminal / command window
     - Mac: `which python`
       `which ipython`
     - Should include “anaconda” or “canopy” in result
  3. In the terminal/command window:

```
pip install --upgrade pip
conda install netCDF4
pip install landlab
```

4. Get a copy of the exercises
   - In a browser, navigate to:
   - https://github.com/landlab/drivers



A brief overview of Python and Numpy

Landlab in action:
Fault-scarp diffusion example

Zhangye thrust fault scarp, Qilian Shan, NE Tibet (Photo © Ralf Hetzel; see Hetzel et al. [2004])

The scientific problem:
How, and how fast, do geomorphic processes modify fault-scarp morphology?

Zhangye thrust fault scarp, Qilian Shan, NE Tibet (Photo © Ralf Hetzel; see Hetzel et al. [2004])

The mathematical problem

$$\boxed{\frac{\partial \eta}{\partial t} = -\nabla \mathbf{q}_s}$$

$\eta =$ land-surface elevation

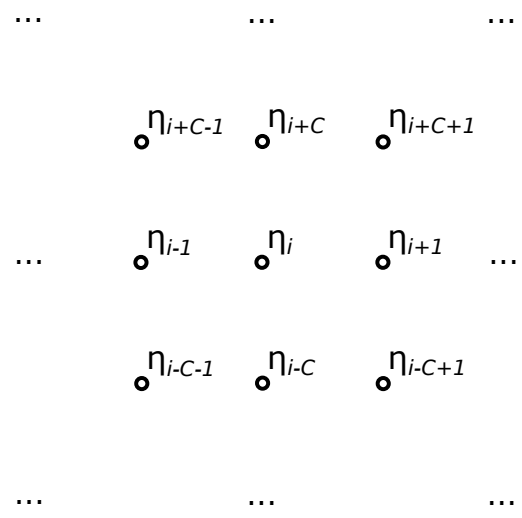
$t =$ time

$\mathbf{q} =$ sediment flux $[L^2/T]$

$$\boxed{\mathbf{q} = -D\nabla\eta}$$

$D =$ transport coefficient $[L^2/T]$

The numerical problem: finite-volume solution scheme



Solve η at a set of **nodes** on a grid

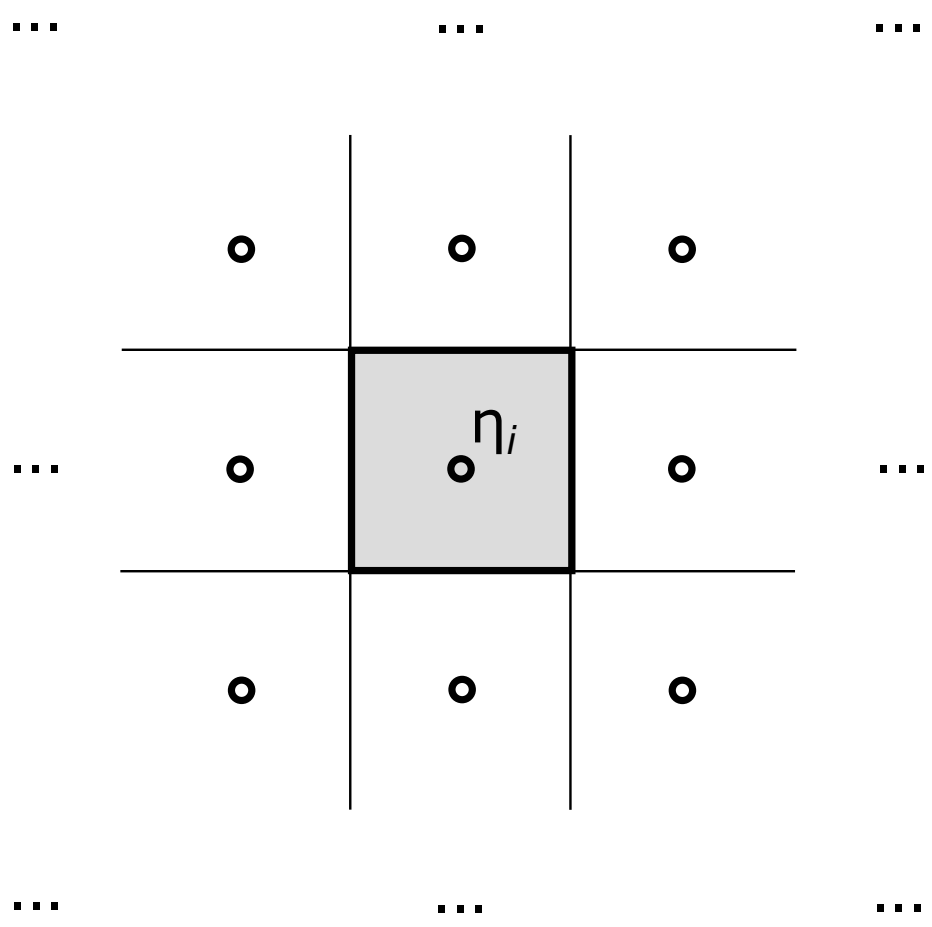
Nodes are numbered $i = 0, 1, 2, ...N - 1$

(note: grid doesn't have to be regular)

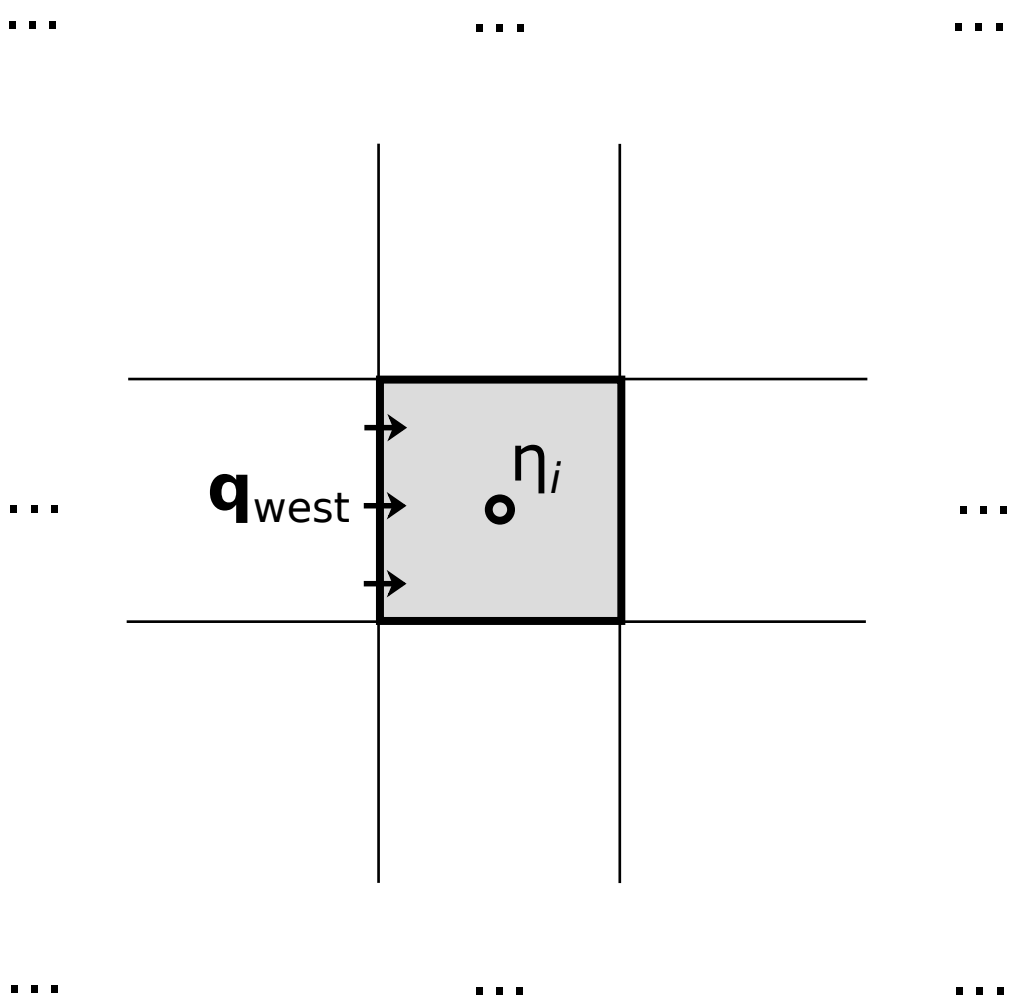
Each interior node i lies within a *cell* whose surface area is A_i.

We can write mass balance for cell i in terms of sediment fluxes across each of its four faces:

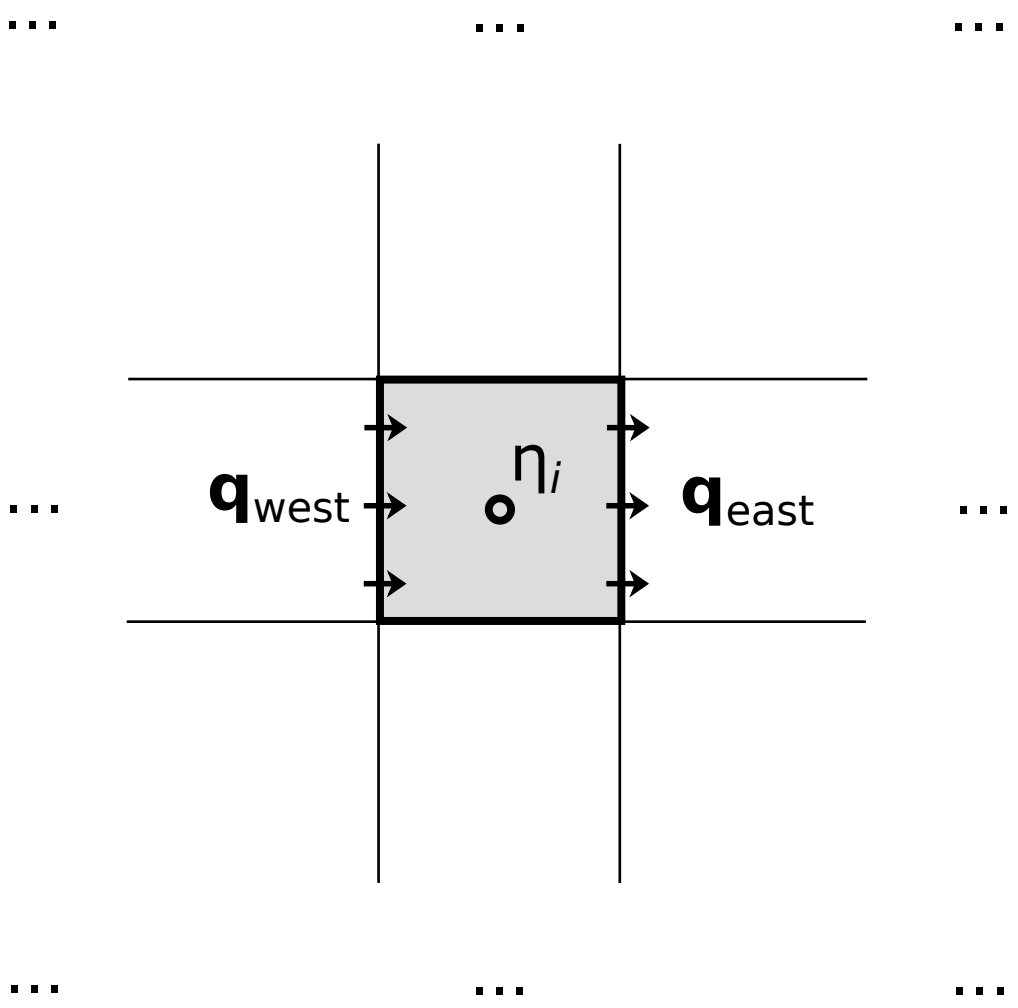
$$\boxed{\frac{d\eta_i}{dt} = \frac{1}{A_i} \sum_{j=1}^{4} \Delta x q_j}$$



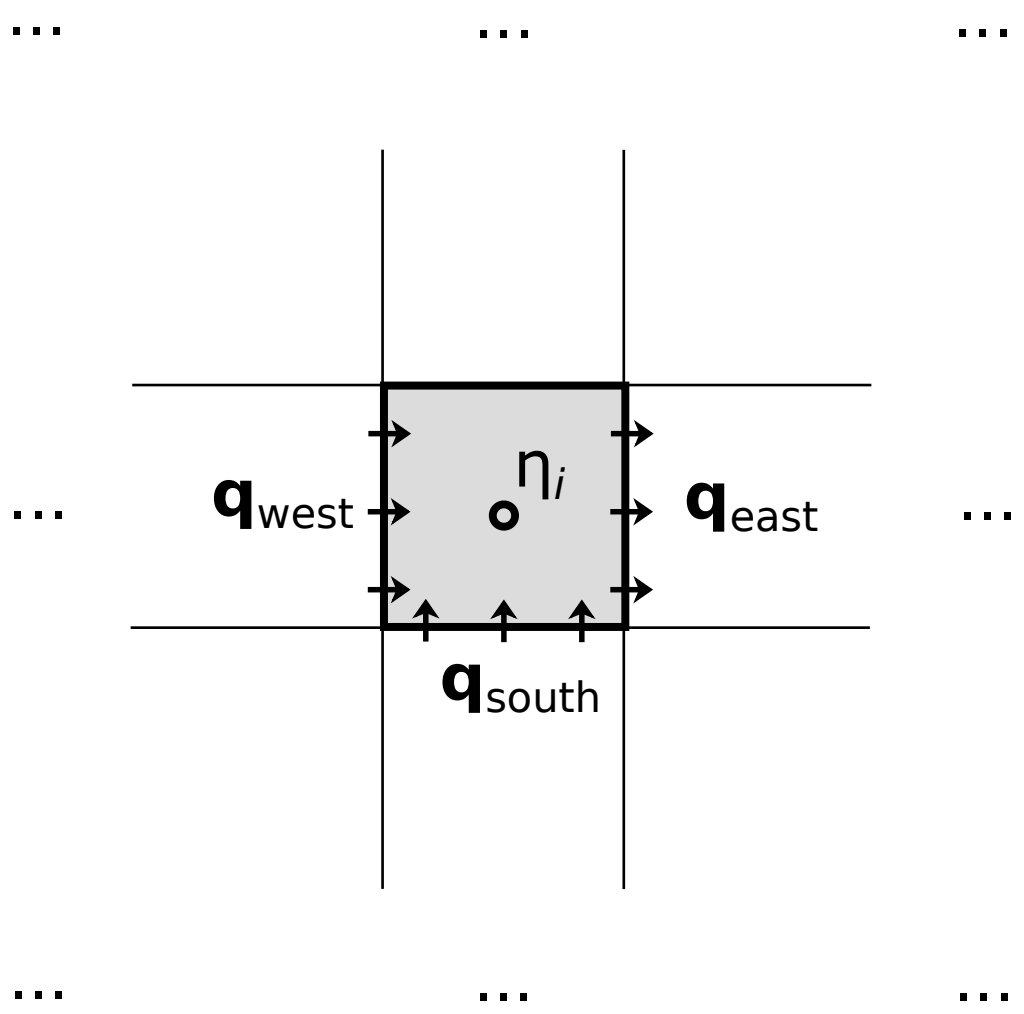
$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\text{west}} \cdots$$



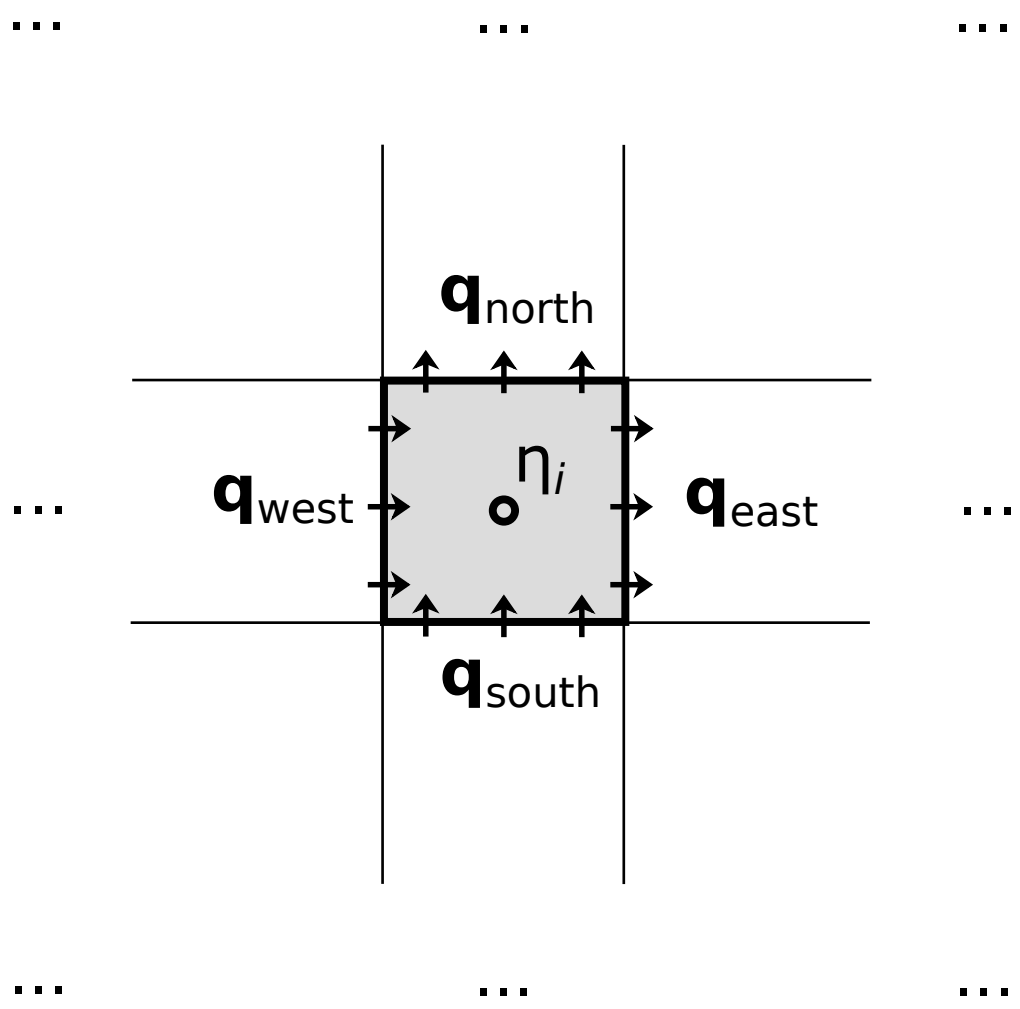
$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\text{west}} - \mathbf{q}_{\text{east}} \cdots$$



$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\text{west}} - \mathbf{q}_{\text{east}} + \mathbf{q}_{\text{south}} \cdots$$

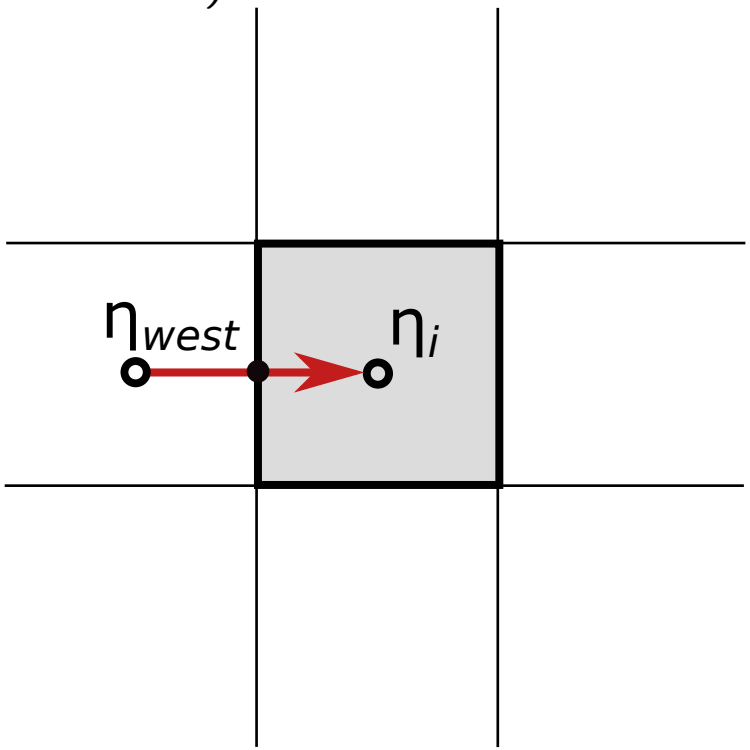


$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\text{west}} + \mathbf{q}_{\text{east}} + \mathbf{q}_{\text{south}} - \mathbf{q}_{\text{north}}]$$



Flux depends on gradient, which is calculated between adjacent nodes:

$$\mathbf{q}_{\text{west}} = -D \left.\frac{\partial \eta}{\partial x}\right|_{\text{(west face)}} \approx -D \left(\frac{\eta_i - \eta_{\text{west}}}{\Delta x}\right)$$



link from η_{west} to η_i

Discretize the time derivative as a forward difference:

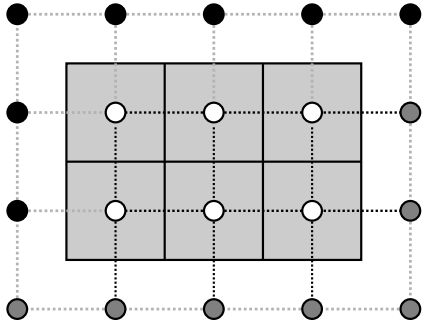
$$\frac{\partial \eta}{\partial t} \approx \frac{\eta_i^{t+1} - \eta_i^t}{\Delta t}$$

Now let's build a little model

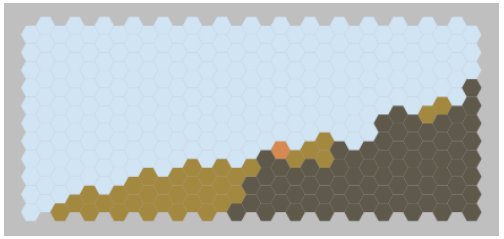
- To follow along, open iPython Notebook "LandlabFaultScarpDemo.ipynb"

Under the hood: Landlab grids

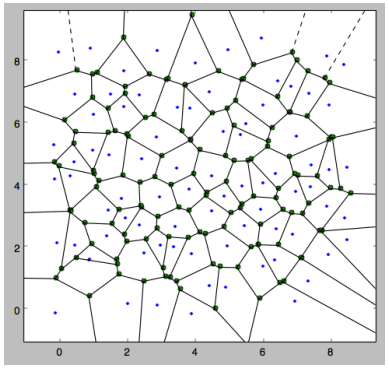
RASTER



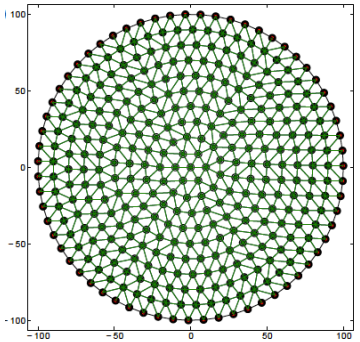
HEXAGONAL



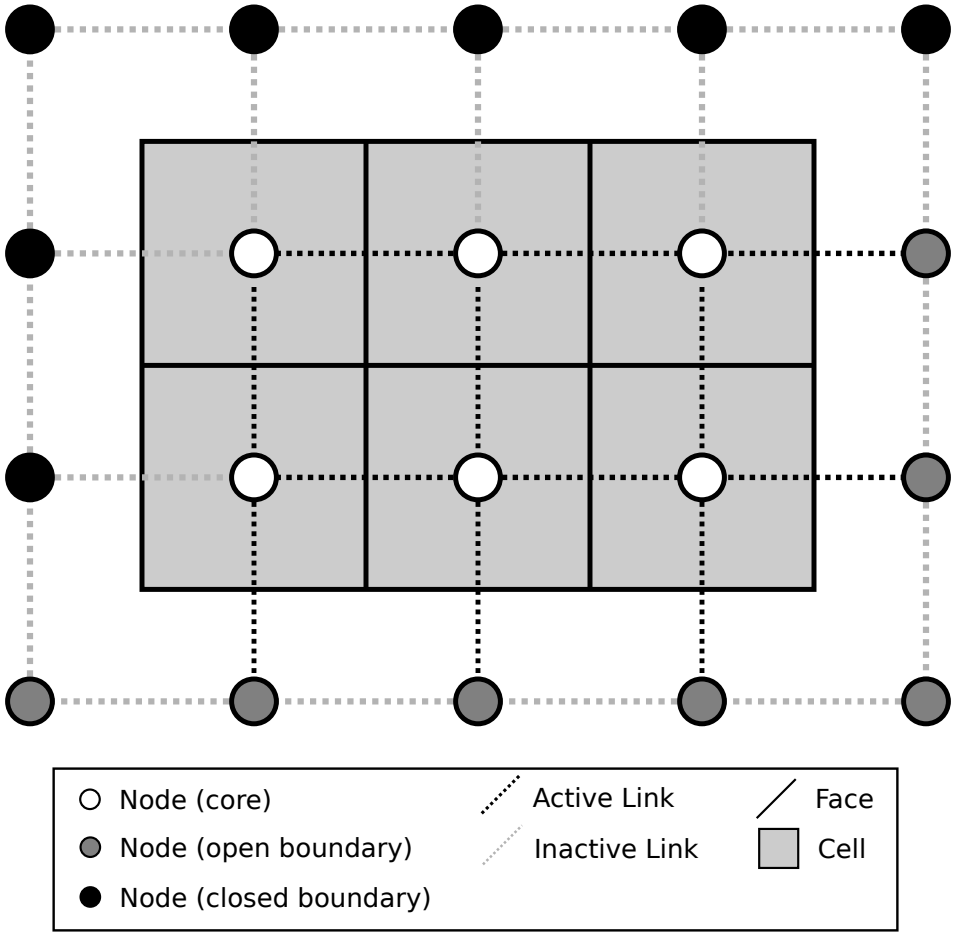
VORONOI / DELAUNAY



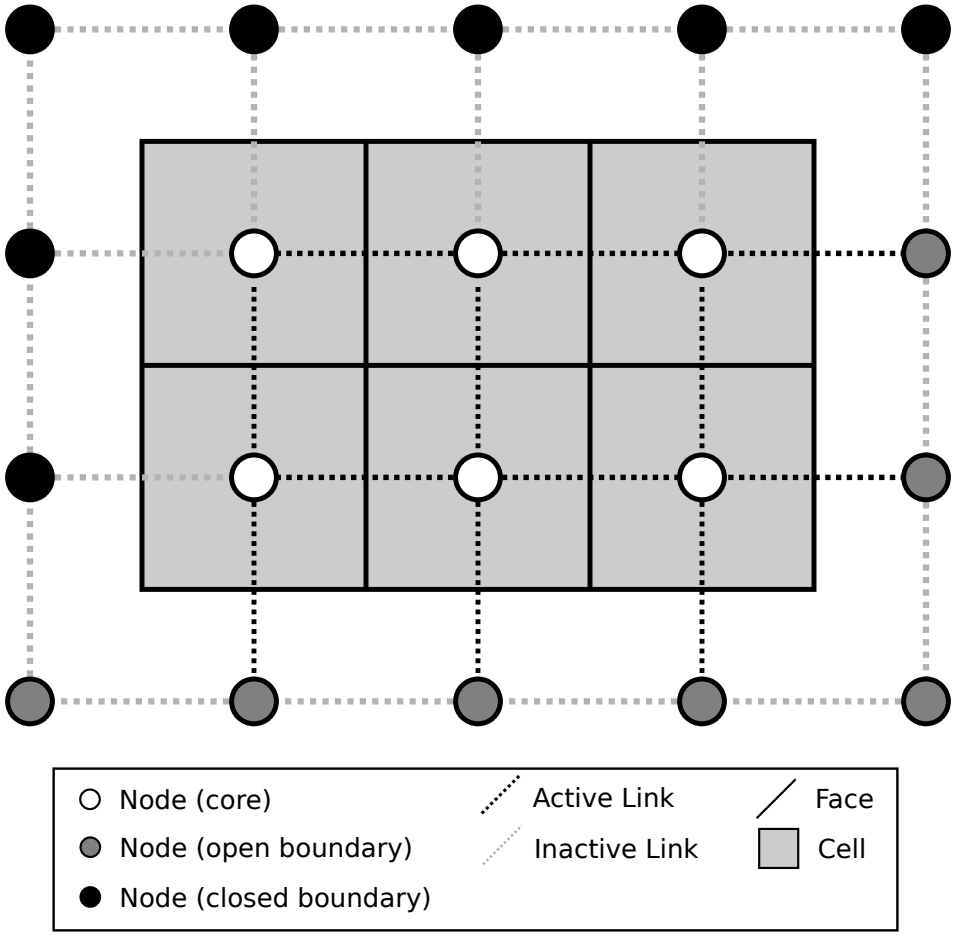
RADIAL



A Landlab grid includes **nodes, links,** and **cells**



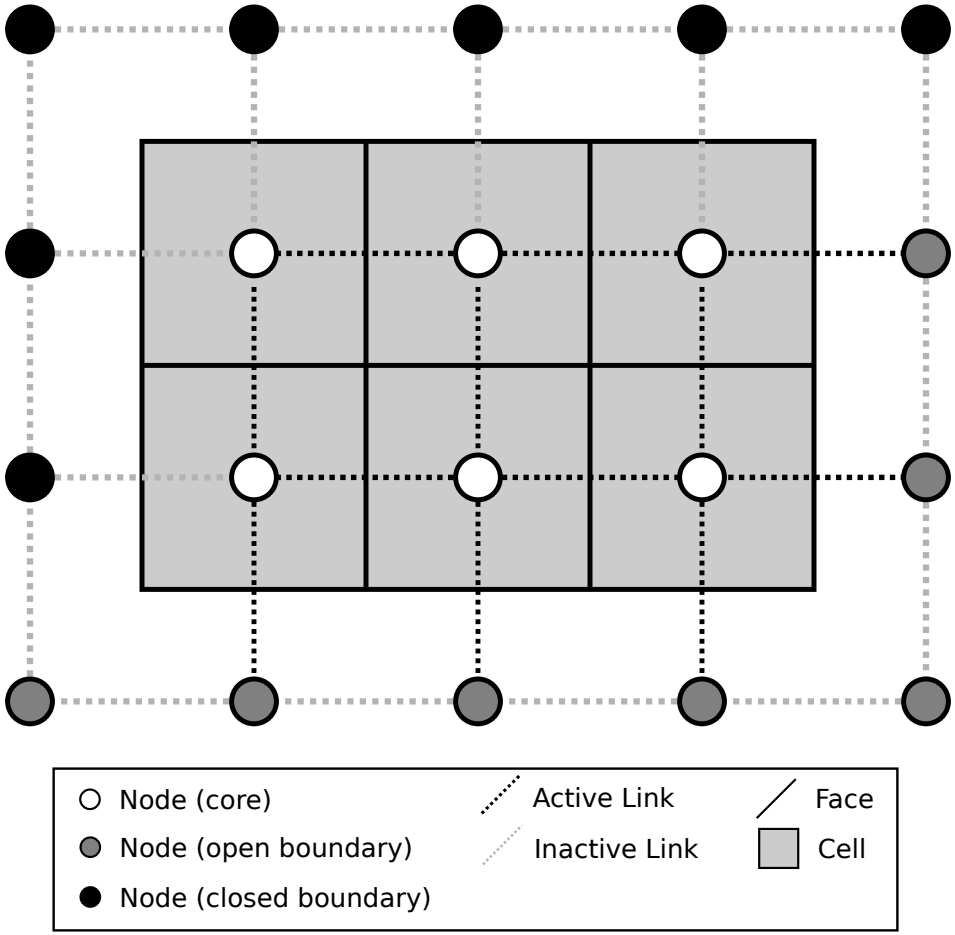
Nodes are classified as **core** or **boundary**
Boundary nodes are either **open** or **closed**



Each interior node lies in a **cell**

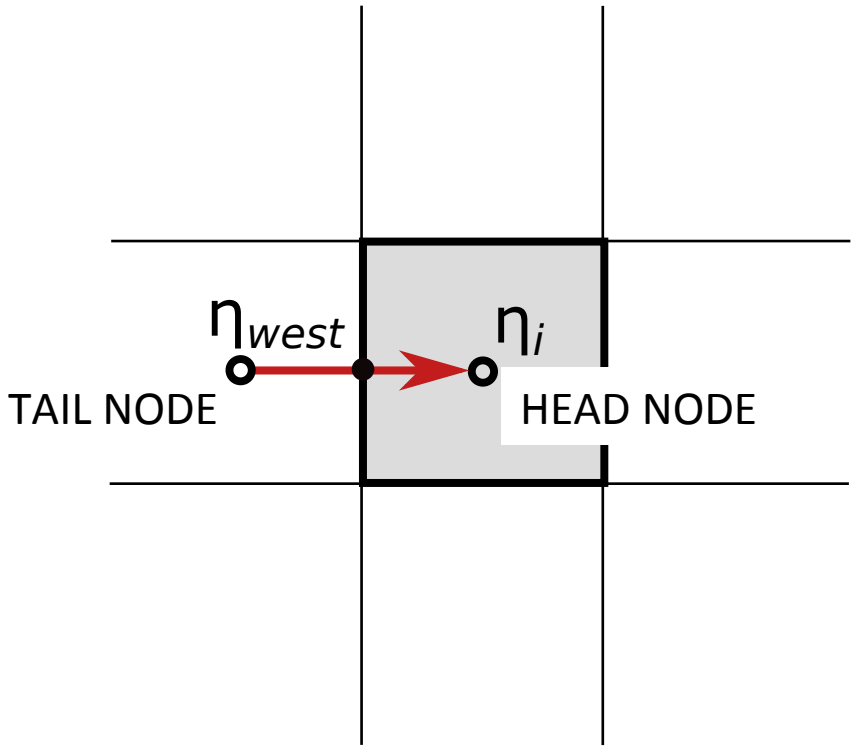
Each cell has a set of **faces**

There is one link for each face



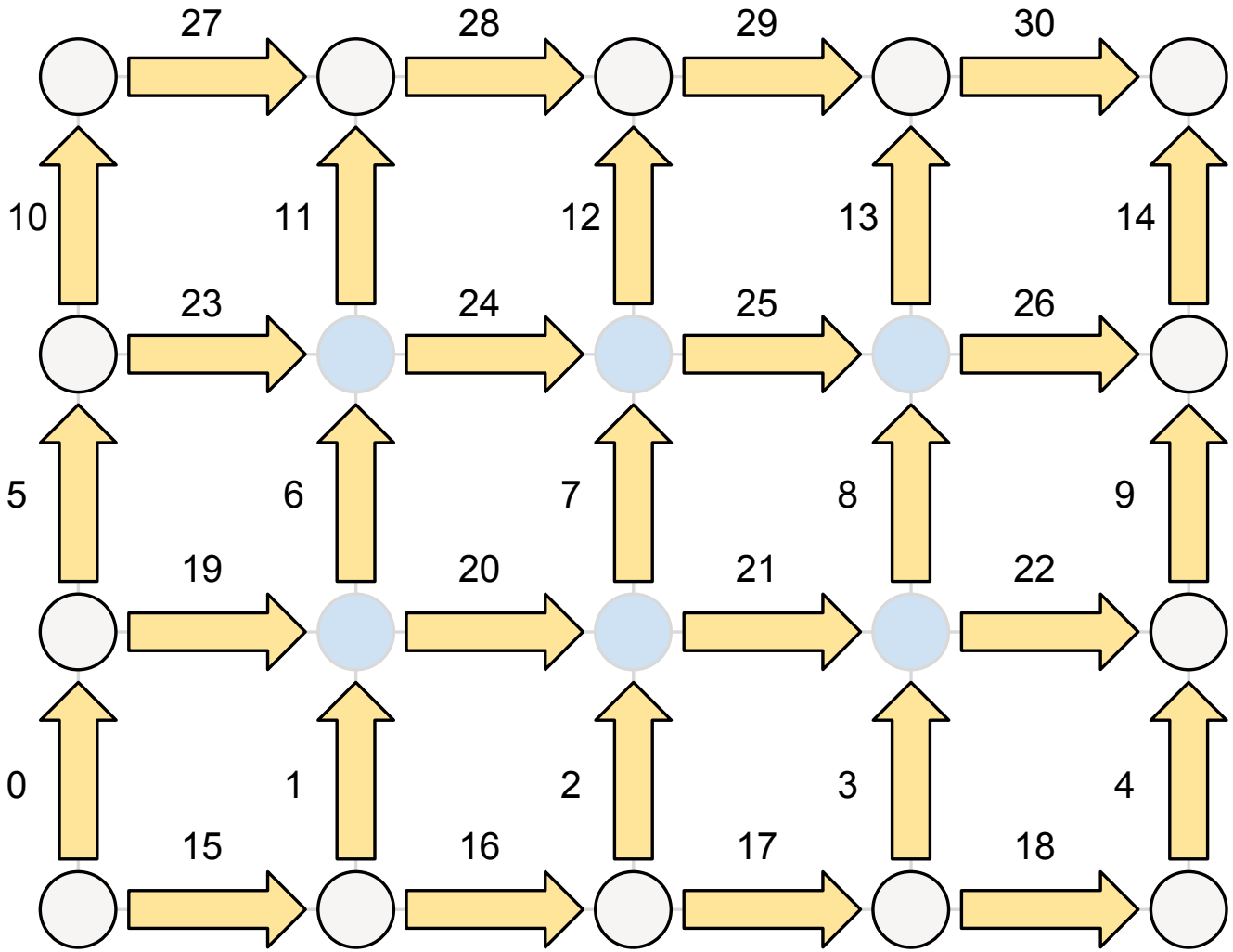
Links

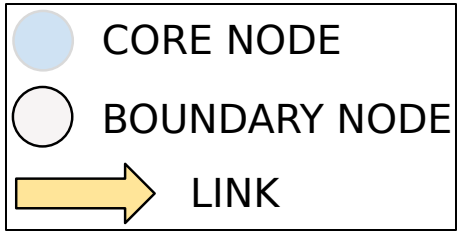
- Each pair of neighboring nodes is connected by a **link**
- Links are directed line segments,
- Direction is from **tail node** to **head node**



link from η_{west} to η_i

Links and nodes

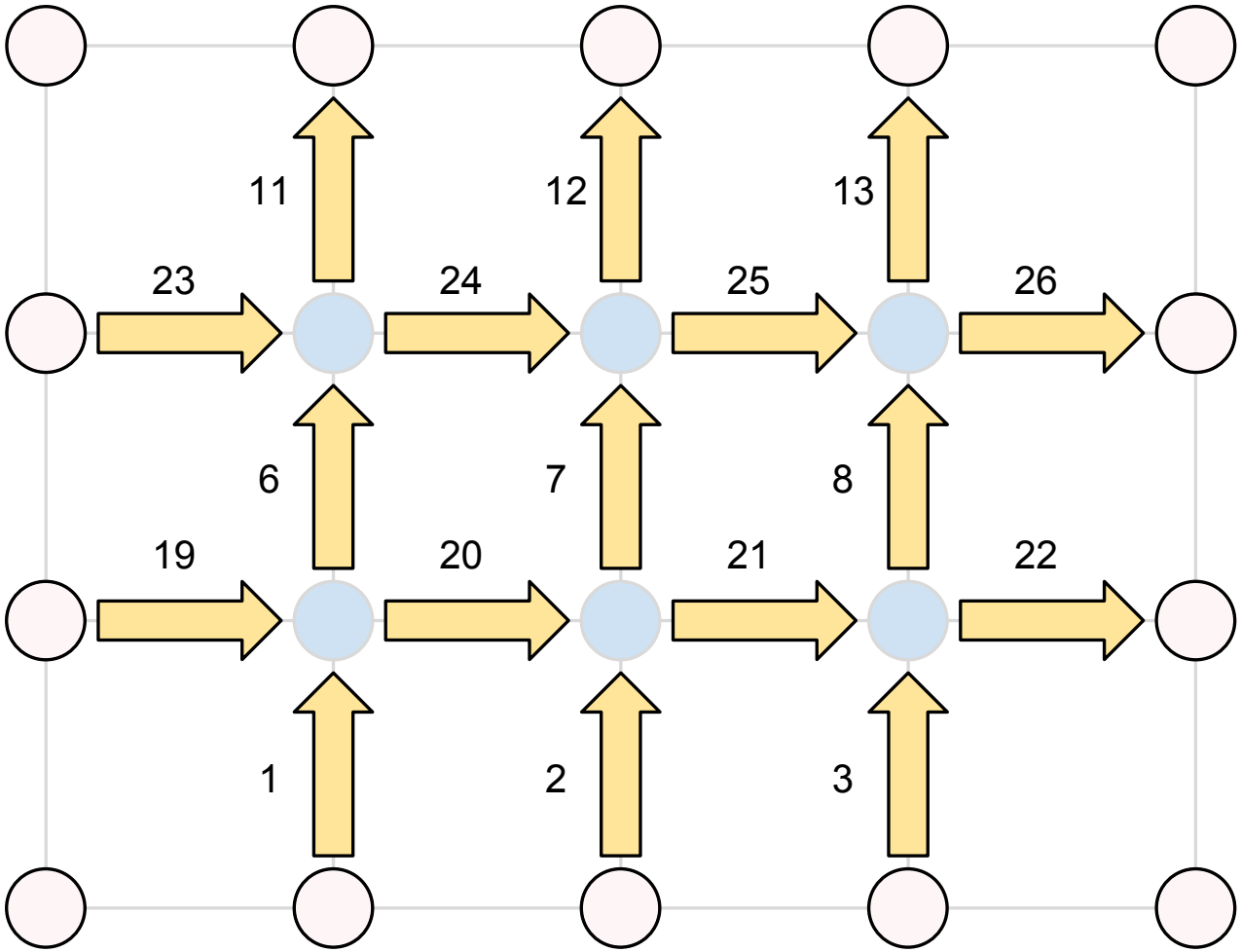


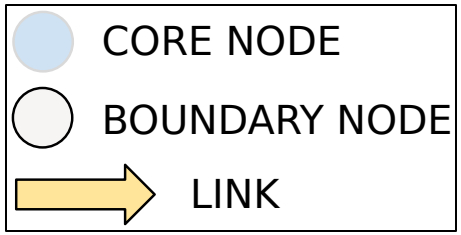


An **active link** connects either:

- Two core nodes
- A core node and an open boundary node

Active links and nodes

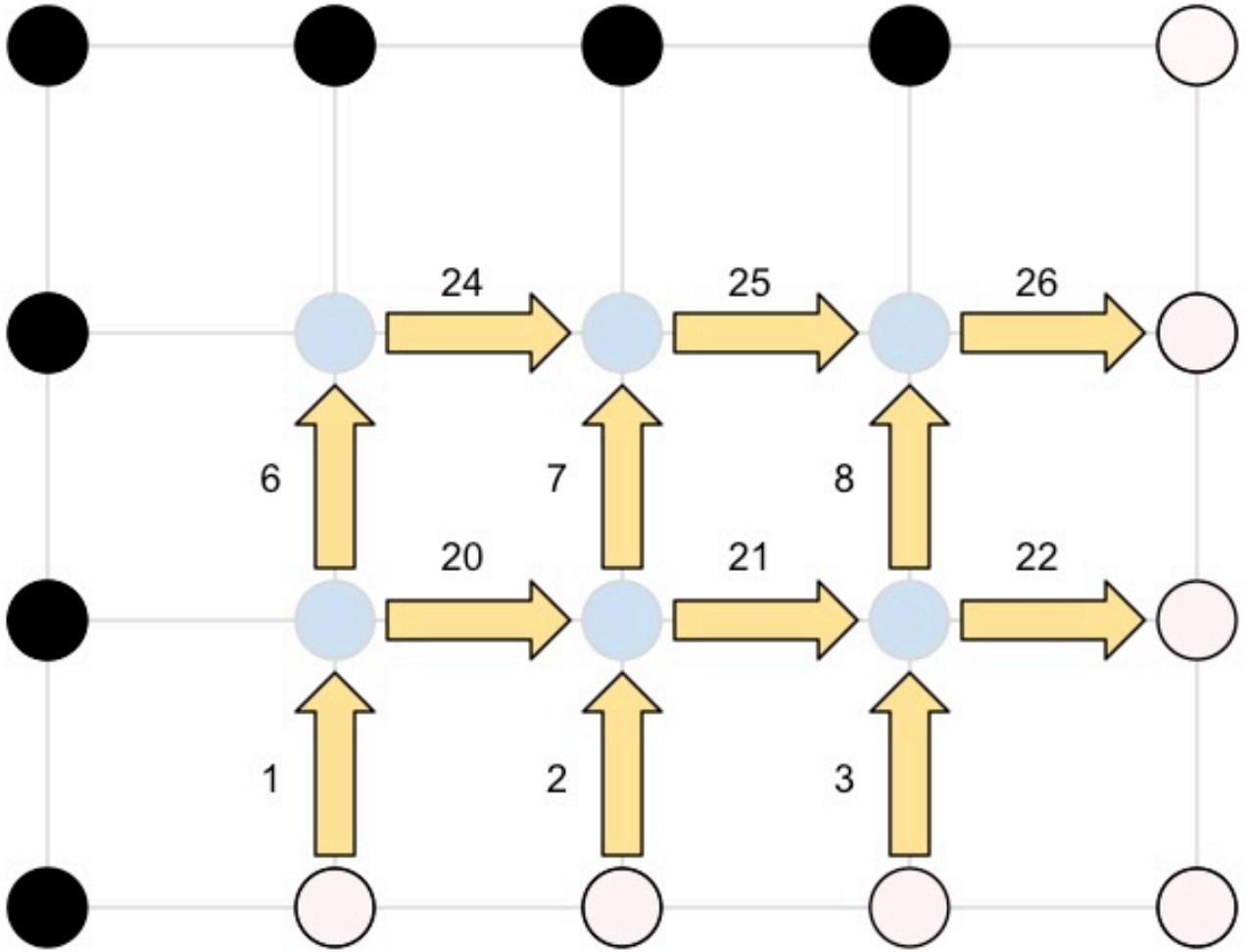


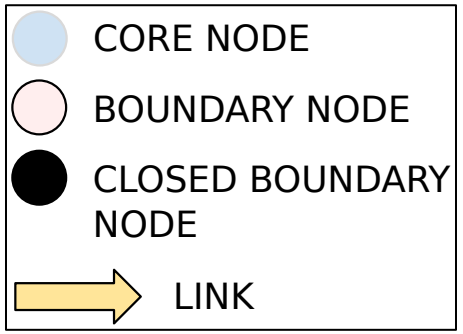


An **active link** connects either:

- Two core nodes
- A core node and an open boundary node

Active links and nodes (with some closed boundary nodes)





An **active link** connects either:

- Two core nodes
- A core node and an open boundary node

The grid object

- The line

```
rg = RasterModelGrid(4, 5, 1.0)
```

- Creates and initializes a *RasterModelGrid object*
- In object-oriented speak, you have created a new *instance* of the `RasterModelGrid` *class*
- Like most objects, `RasterModelGrid` includes both data and functions (a.k.a., *methods*)
- You can attach *data fields* to the grid

Example: creating and working with a grid and its fields

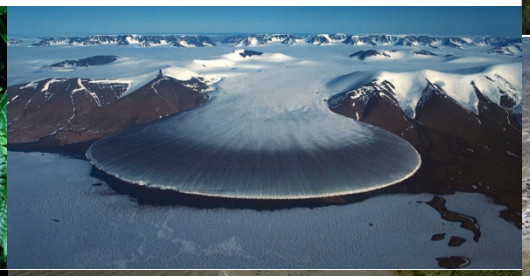
- To follow along, open iPython Notebook "grid_object_demo.ipynb"

The components

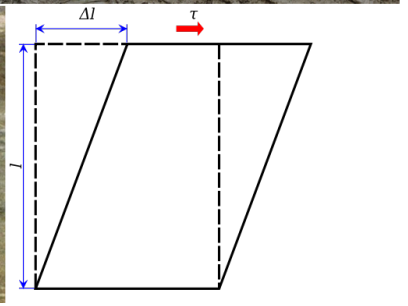
- Describe individual surface processes
- “Plug & Play”
- Standard interface
- Use the library, or BYO

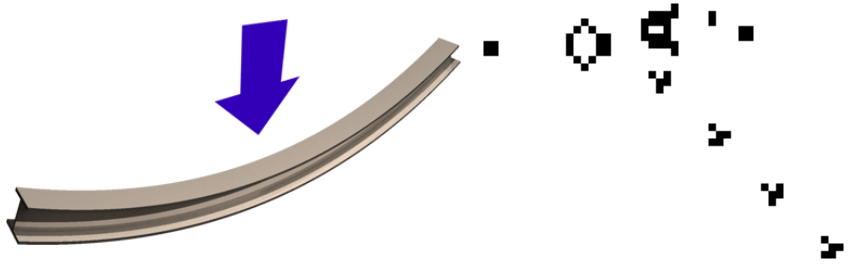








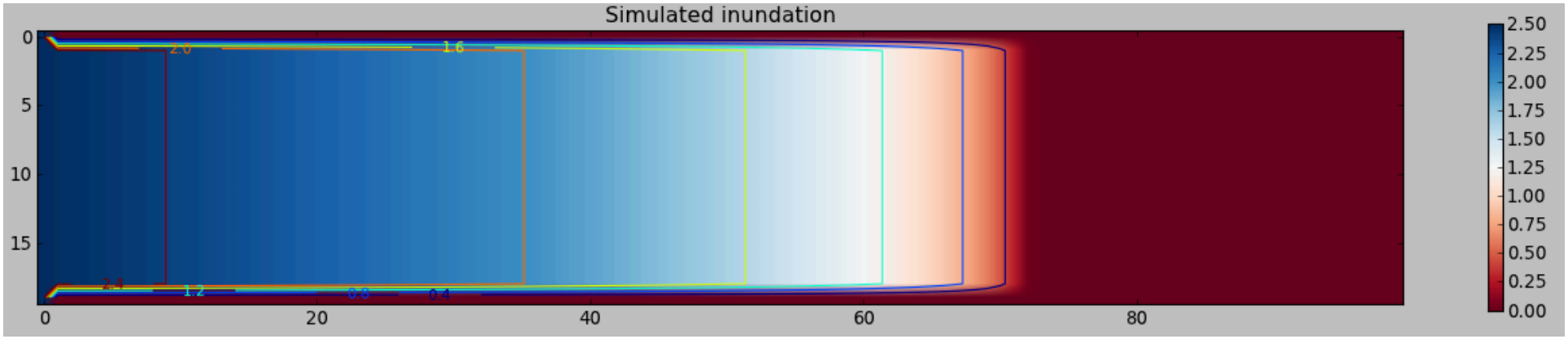


Using components: a simple demo

- To follow along, open iPython Notebook “component_tutorial.ipynb”

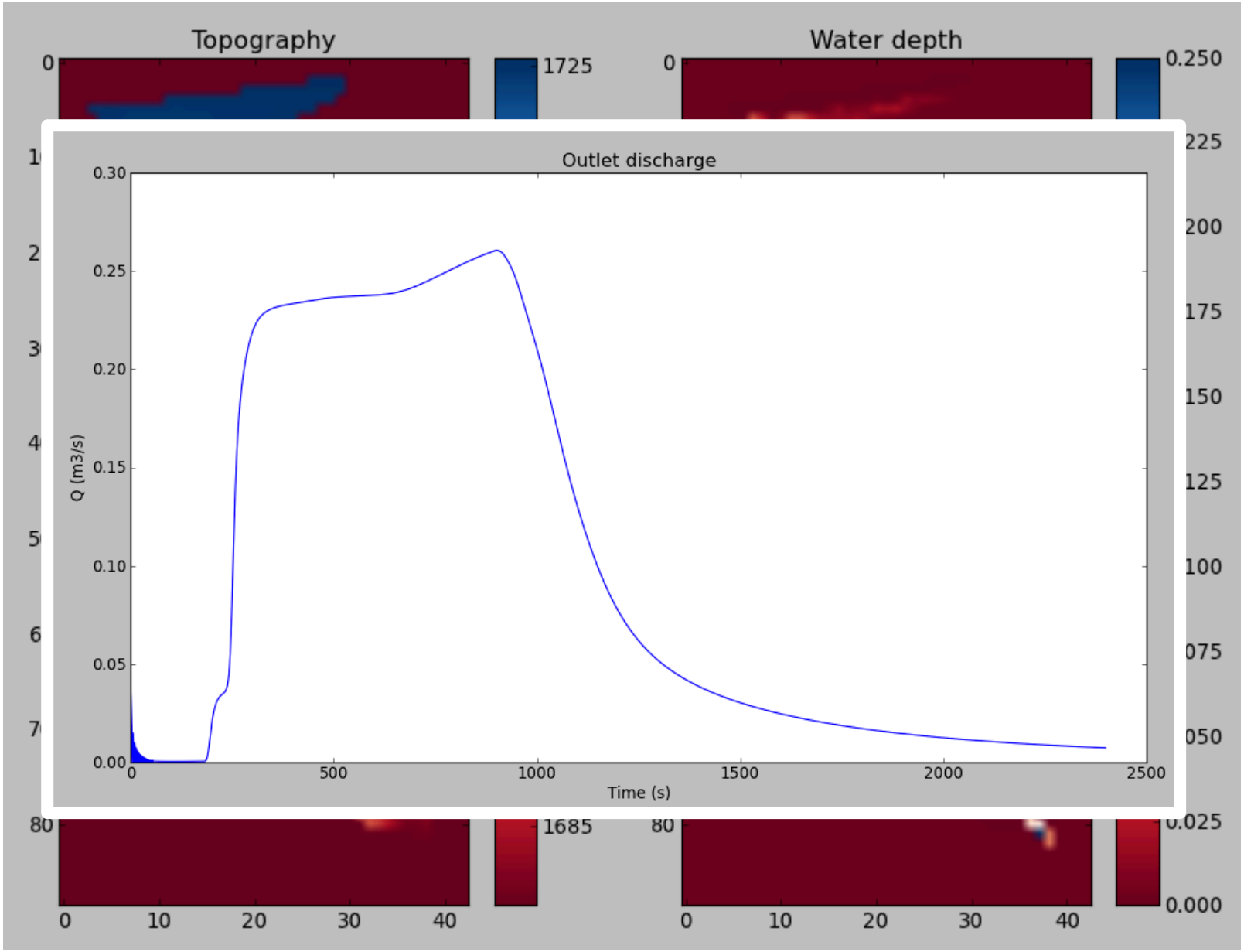
A glimpse at some existing components

Flood-wave propagation

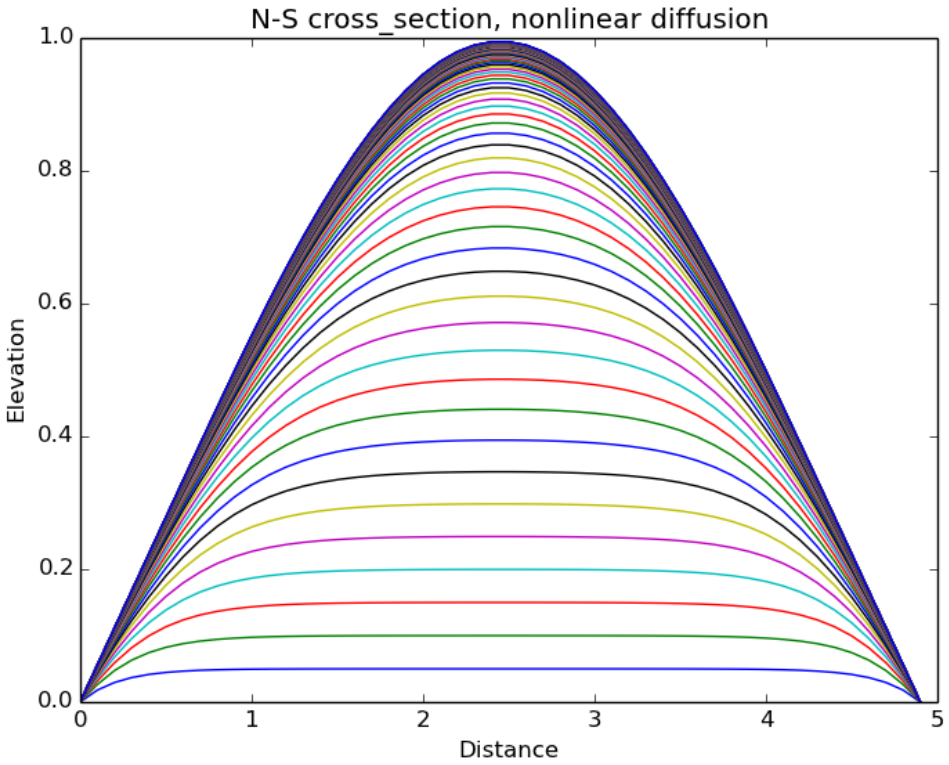


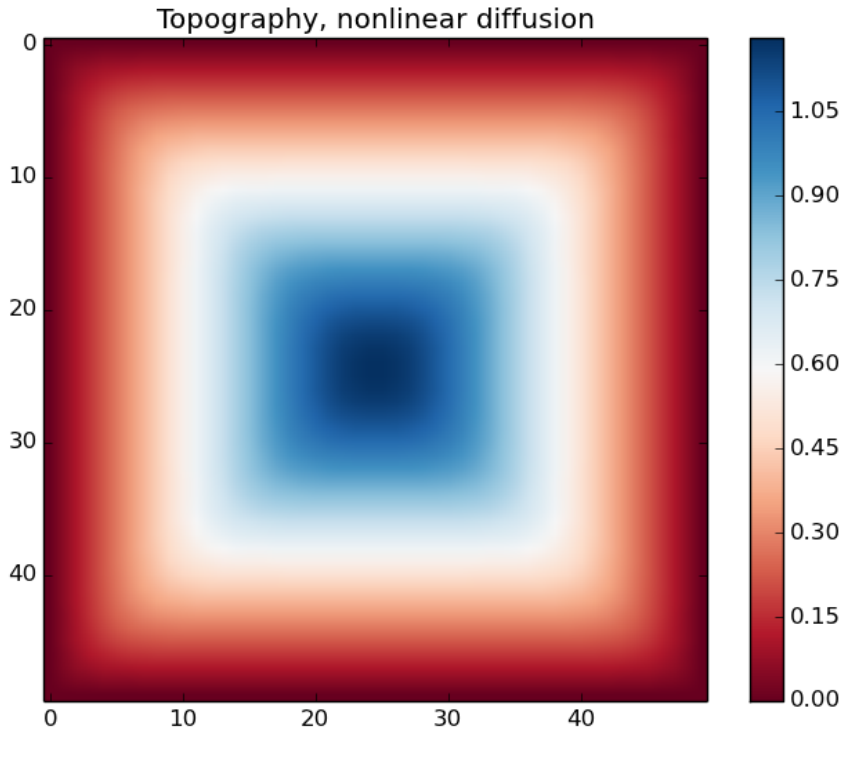
- Follows math laid out in Bates et al. (2010)
- Calculates flux divergences at nodes
- Very simple to implement in driver

Overland flow on a DEM

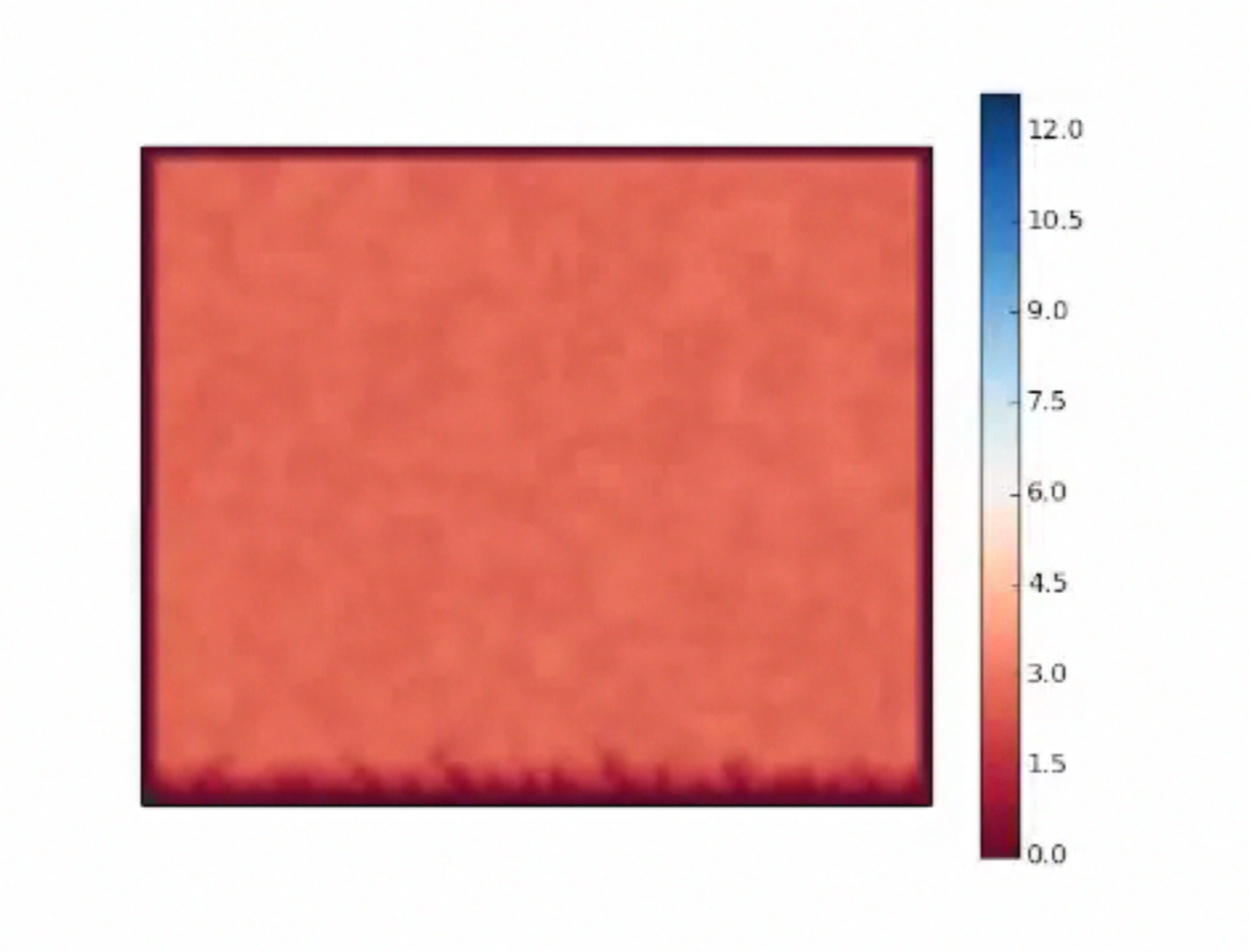


Nonlinear diffusion

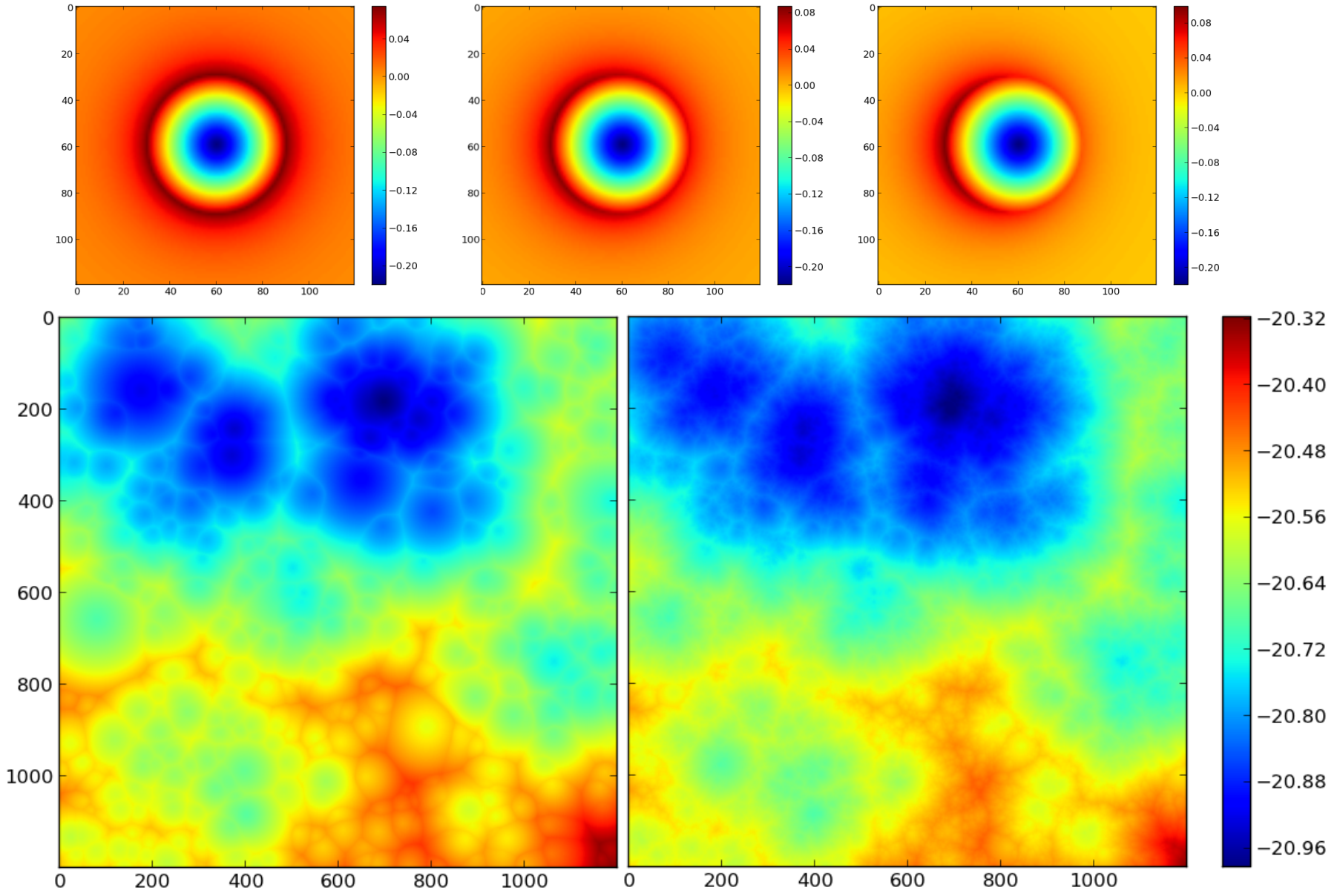




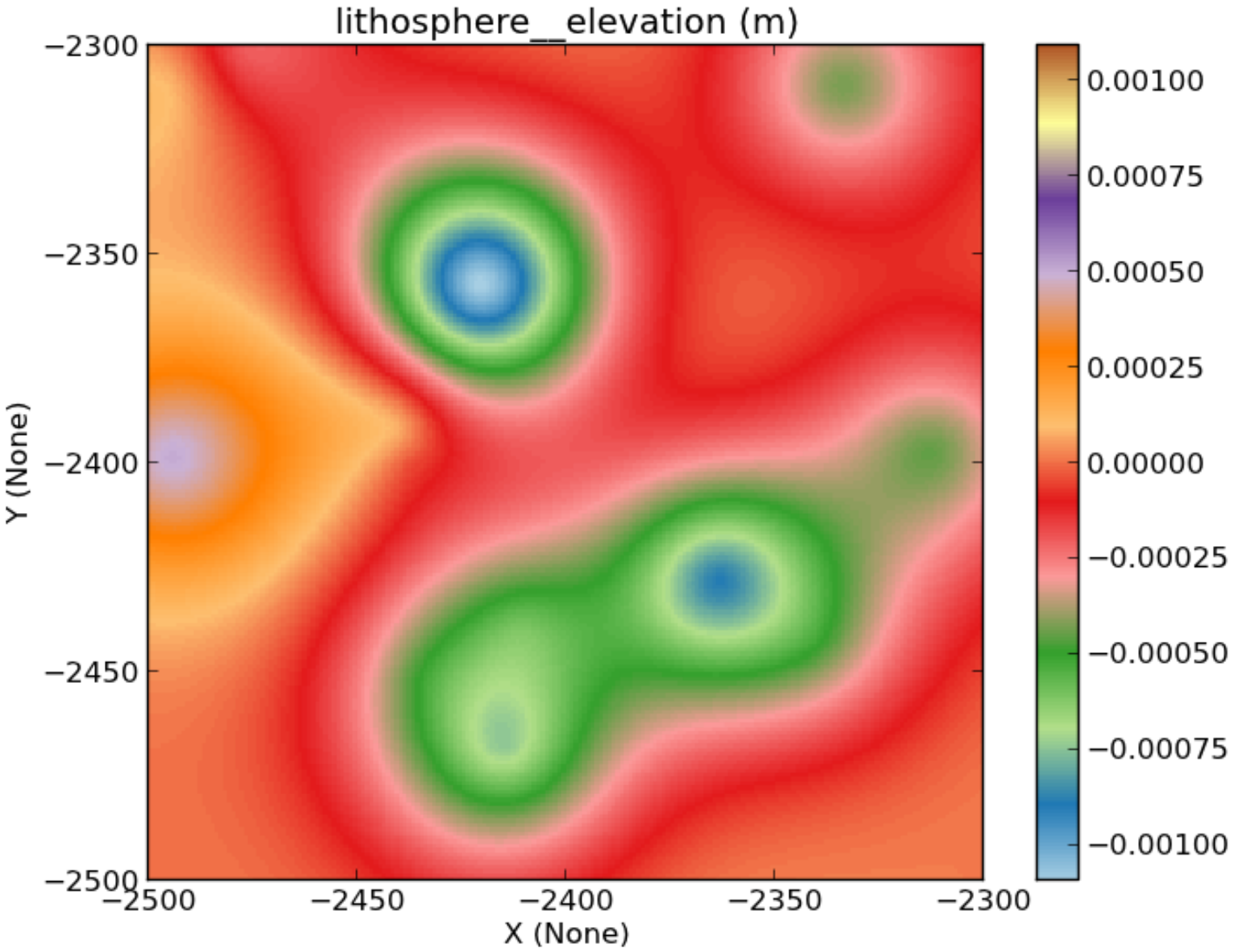
Stream power



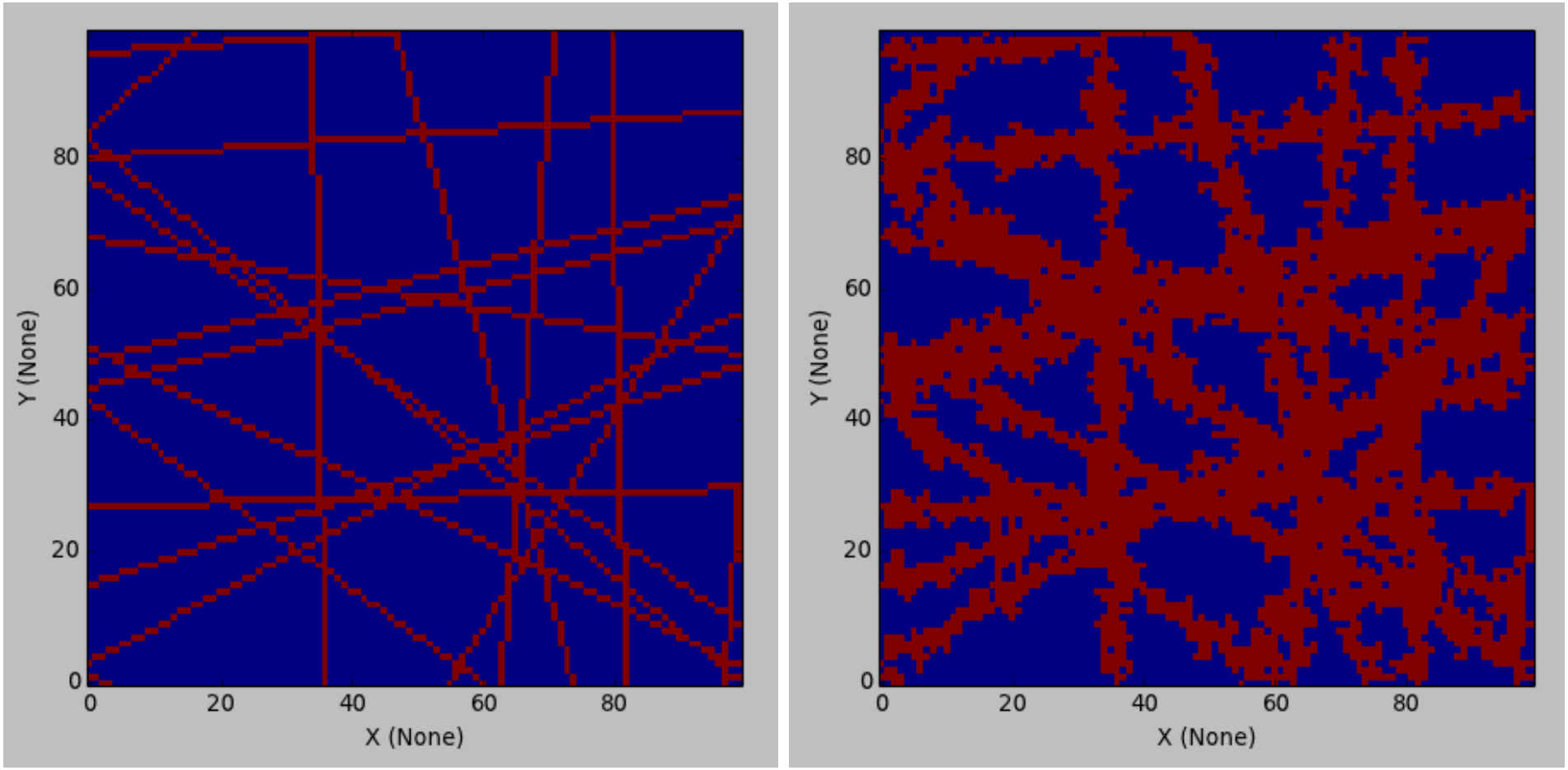
Impact cratering



Flexure


lithosphere__elevation (m)
−2300
−2350
−2400
−2450
−2500
Y (None)
−2500
−2450
−2400
−2350
−2300
X (None)
0.00100
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−0.00100

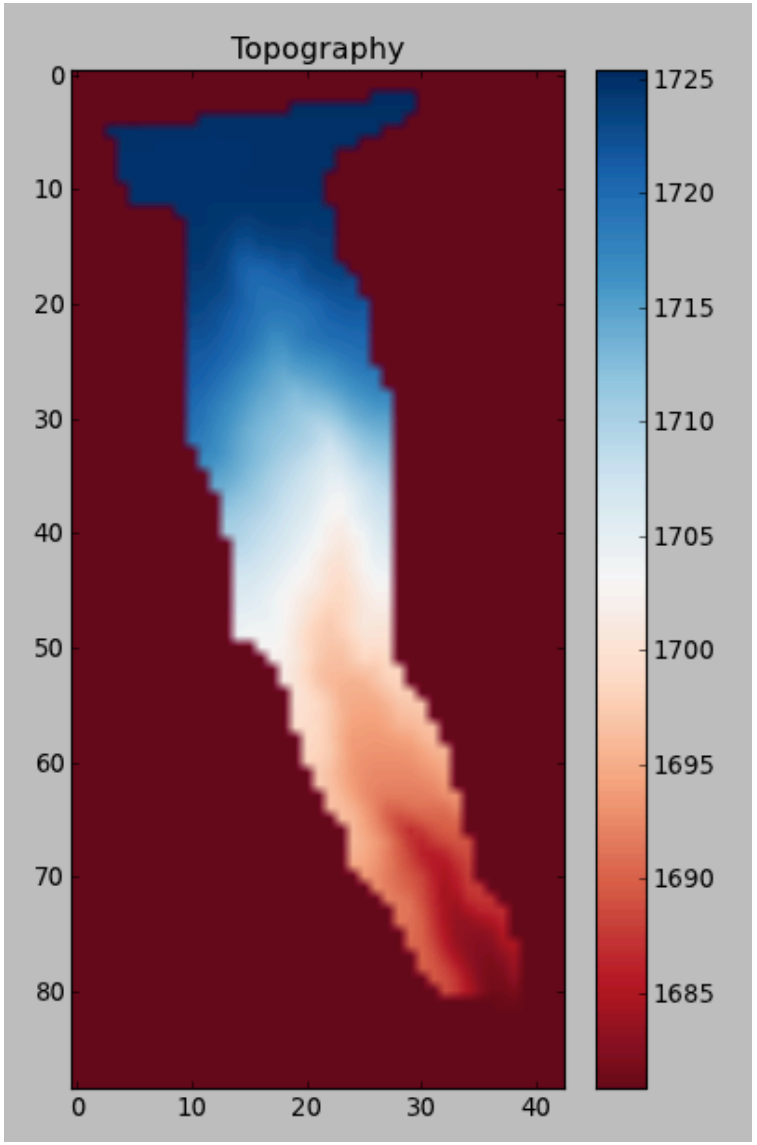
Cellular automata



Simple CA model of weathering around fractures (red) into a medium (blue).

The utilities

- Read DEMs
- Read .txt
- Output routines (e.g., netCDF)
- Visualization tools
- Other repetitive tasks



Tutorials and additional examples

- Fault scarp example, with and without components:
  - http://landlab.readthedocs.org/
  - Select *"A Tutorial on Quickly Building 2D Models in Landlab"*
- Other documentation and tutorials on the main documentation page