Overview of the Landlab Toolkit

CSDMS Webinar, September 14, 2018

Presented by Greg Tucker

With help from the Landlab team: Katherine R. Barnhart[1], Jordan Adams[1,3], Christina Bandaragoda[2], Nicole M. Gasparini[3], Daniel E.J. Hobley[4], Eric Hutton[1], Erkan Istanbulluoglu[2], Jenny Knuth[1], Nathan Lyons[3], Margaux Mouchene[1,3], Sai Siddhartha Nudurupati[2]

1 – University of Colorado, Boulder

2 – University of Washington

3 – Tulane University

4 – Cardiff University

Upcoming CSDMS Webinars

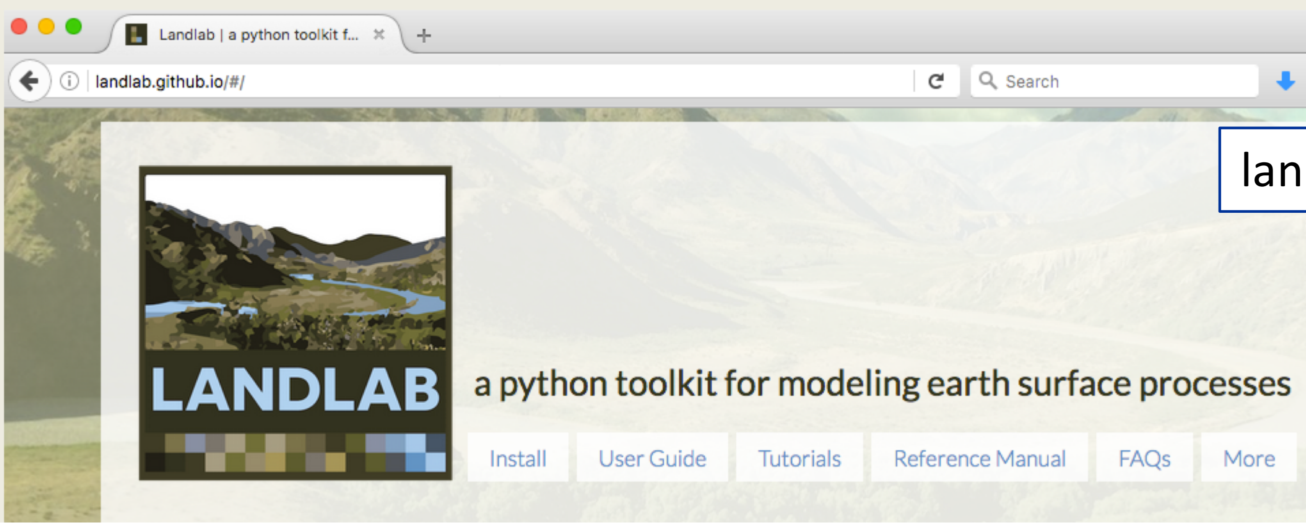
- **Using CSDMS in the Classroom** - Learn about CSDMS software for running a suite of earth surface models through a web-based modeling tool (WMT). This webinar will share improved ways of using this tool in the classroom, gives a quick reminder demo, and points in detail to the resources online. Instructor: **Irina Overeem**, CSDMS Deputy Director, University of Colorado, Boulder - October 9th, 12PM Eastern Time - Register

  CSDMS Basic Model Interface (BMI) - When equipped with a Basic Model Interface, a model is given a common set of functions for configuring and running the model (as well as getting and setting its state). Models with BMIs can communicate with each other and be coupled in a modeling framework. The coupling of models from different authors in different disciplines may open new paths to scientific discovery. In this first of a set of webinars on the CSDMS BMI, we'll provide an overview of BMI and the functions that define it. This webinar is appropriate for new users of BMI, although experienced users may also find it useful. Instructor: **Mark Piper**, Research Software Engineer, University of Colorado, Boulder - November 13th, 12PM Eastern Time – Register

CSDMS
COMMUNITY SURFACE DYNAMICS MODELING SYSTEM

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
- Companion to CSDMS Python Modeling Tool (PyMT)

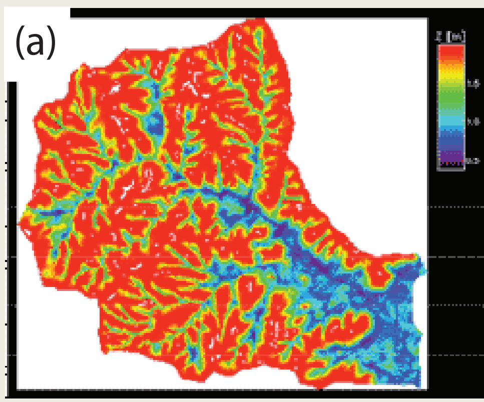


landlab.github.io

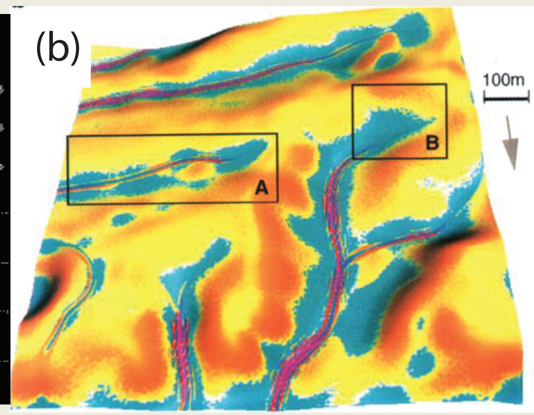
Spatially distributed (2D) process models are vital for studying earth's surface

… and the models some interesting things in common

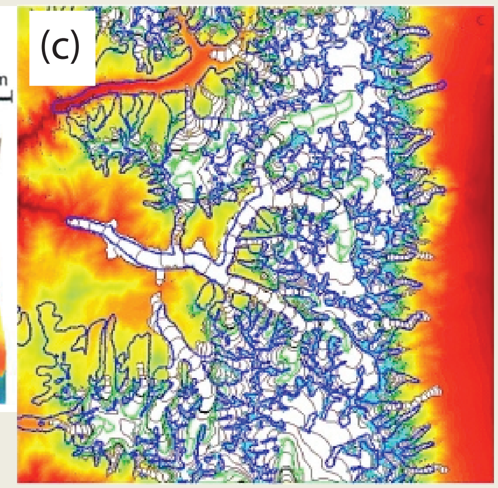
CATCHMENT HYDROLOGY
(Ivanov et al., 2004)

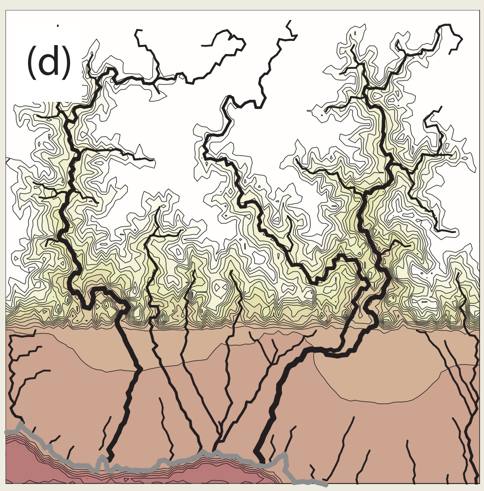


SOIL EROSION
(Mitas and Mitasova, 1998)

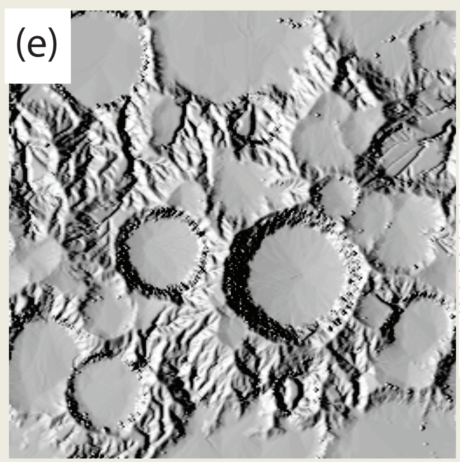


GLACIER DYNAMICS
(Kessler et al., 2006)

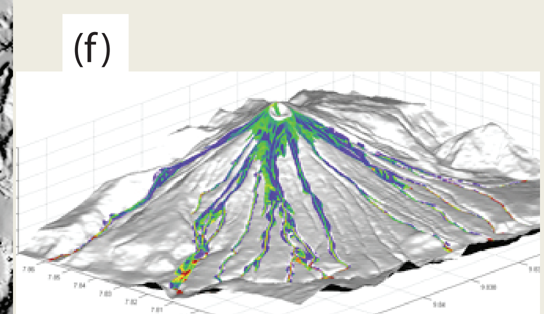




LANDSCAPE EVOLUTION
(Tucker and Hancock, 2010)

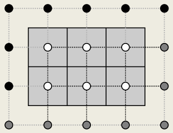


IMPACT CRATERING AND DEGRADATION
(Howard, 2007)



LAVA FLOWS
(Kelfoun et al., 2009)

Design goals



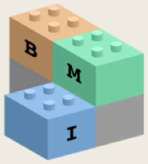
Gridding: set up 2D grid in a few lines of code

Data: ”attach” data layers to grid elements

Plug & Play: reusable, standardized components

Light interface: easy to learn and use

Housekeeping: common tasks (I/O, basic plotting)

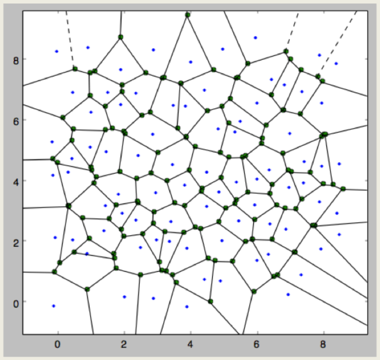


Integration: use in other frameworks

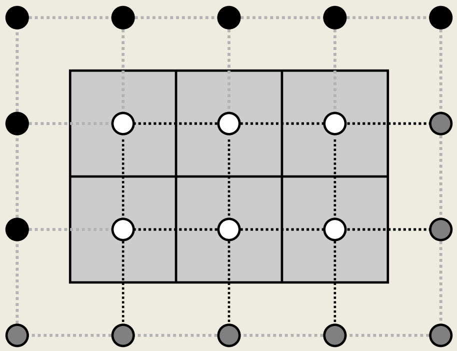
Landlab: grid management

- Grids are objects
- Grids use flat arrays
- Data *fields* can be attached to grid elements
- Multiple grid types
- Built-in numerical functions:
  - gradient
  - divergence

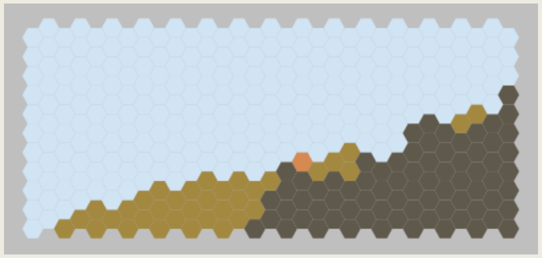
VORONOI / DELAUNAY



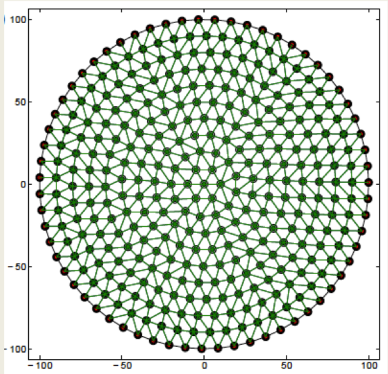
RASTER

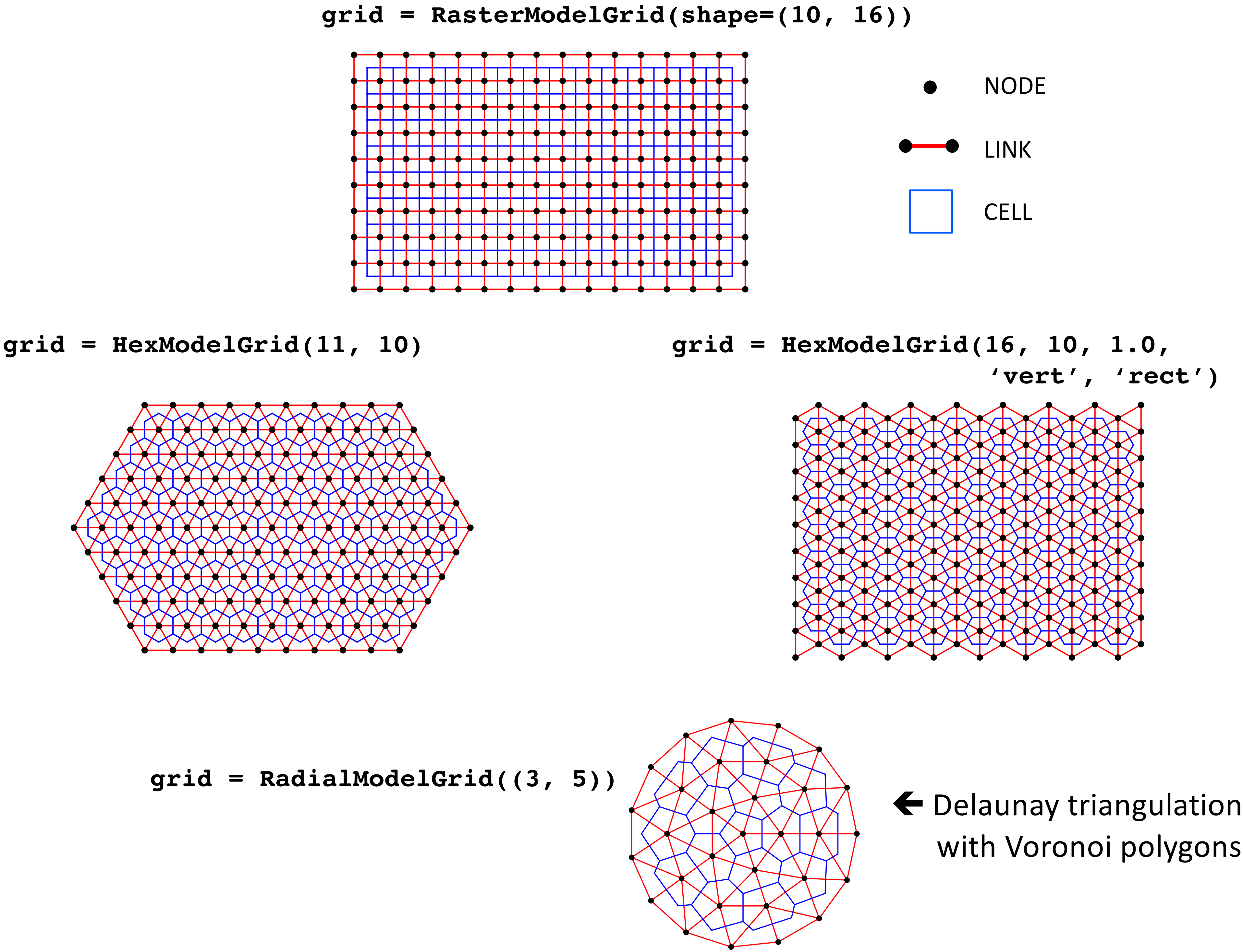


HEXAGONAL



RADIAL





grid = RasterModelGrid(shape=(10, 16))
NODE
LINK
CELL
grid = HexModelGrid(11, 10)
grid = HexModelGrid(16, 10, 1.0,
'vert', 'rect')
grid = RadialModelGrid((3, 5))
Delaunay triangulation
with Voronoi polygons

Components: plug-and-play capability

Components: plug-and-play capability

COMPONENTS ARE PYTHON CLASSES →

```
class HeatDiffuser(Component):
```

Components: plug-and-play capability

COMPONENTS ARE PYTHON CLASSES →

METADATA (bracket spanning the class attributes)

```
class HeatDiffuser(Component):

    _name = 'HeatDiffuser'

    _input_var_names = 'temperature'

    _output_var_names = ('temperature', 'temperature__gradient', 'heat__flux')

    _var_units = { 'temperature' : 'o',
                   'temperature__gradient' : 'o m**-1',
                   'heat__flux' : 'W m**-2' }

    _var_mapping = { 'temperature' : 'node',
                     'temperature__gradient' : 'link',
                     'heat__flux' : 'link' }

    _var_doc = { 'temperature' : 'Temperature at nodes',
                 'temperature__gradient' : 'Temperature gradient along links',
                 'heat__flux' : 'Heat flux along links' }
```

Components: plug-and-play capability

COMPONENTS ARE PYTHON CLASSES →

METADATA

INITIALIZE →

```python
class HeatDiffuser(Component):

    _name = 'HeatDiffuser'

    _input_var_names = 'temperature'

    _output_var_names = ('temperature', 'temperature__gradient', 'heat__flux')

    _var_units = { 'temperature' : 'o',
                   'temperature__gradient' : 'o m**-1',
                   'heat__flux' : 'W m**-2' }

    _var_mapping = { 'temperature' : 'node',
                     'temperature__gradient' : 'link',
                     'heat__flux' : 'link' }

    _var_doc = { 'temperature' : 'Temperature at nodes',
                 'temperature__gradient' : 'Temperature gradient along links',
                 'heat__flux' : 'Heat flux along links' }

    def __init__(self, grid, thermal__diffusivity=1.0e-6):

        pass  # CREATE FIELDS AND DO OTHER INITIALIZATION HERE
```

Components: plug-and-play capability

COMPONENTS ARE PYTHON CLASSES →

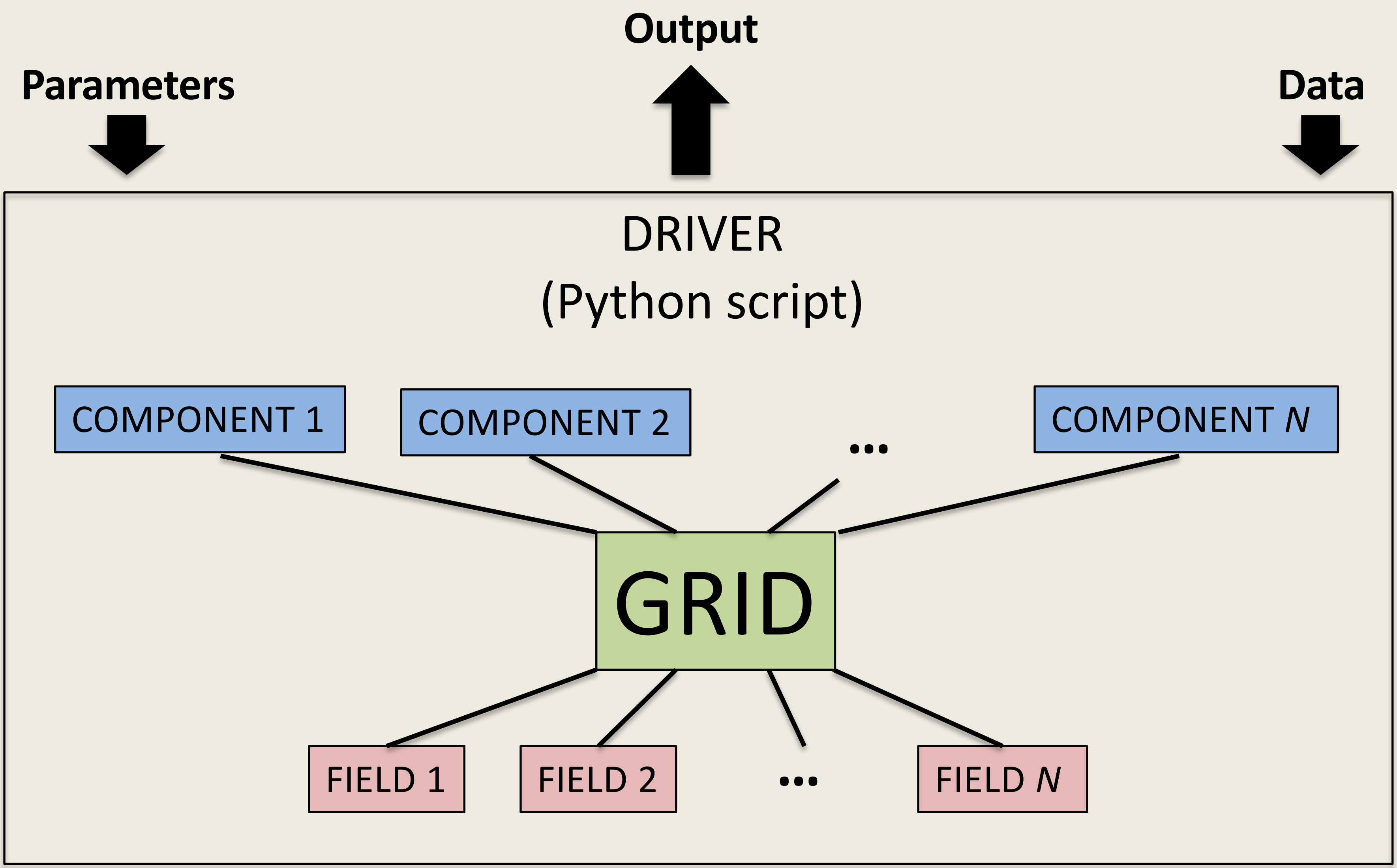
METADATA

INITIALIZE →

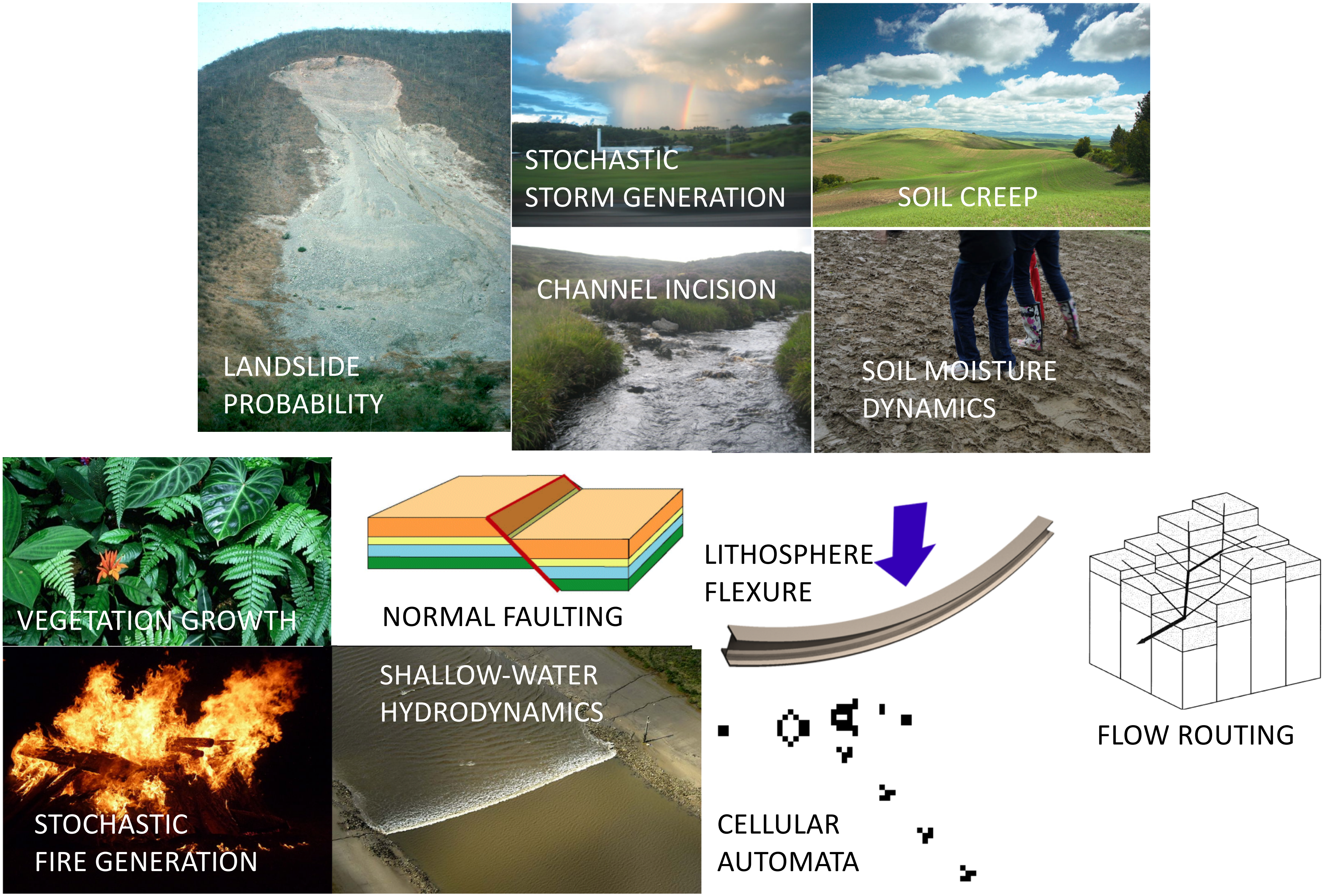
RUN →

```
class HeatDiffuser(Component):

    _name = 'HeatDiffuser'

    _input_var_names = 'temperature'

    _output_var_names = ('temperature', 'temperature__gradient', 'heat__flux')

    _var_units = { 'temperature' : 'o',
                   'temperature__gradient' : 'o m**-1',
                   'heat__flux' : 'W m**-2' }

    _var_mapping = { 'temperature' : 'node',
                     'temperature__gradient' : 'link',
                     'heat__flux' : 'link' }

    _var_doc = { 'temperature' : 'Temperature at nodes',
                 'temperature__gradient' : 'Temperature gradient along links',
                 'heat__flux' : 'Heat flux along links' }

    def __init__(self, grid, thermal__diffusivity=1.0e-6):

        pass  # CREATE FIELDS AND DO OTHER INITIALIZATION HERE

    def run_one_step(self, dt):  # dt IS TIME-STEP DURATION

        pass  # UPDATE GRADIENT, FLUX, AND TEMPERATURE FIELDS FOR NEW TIMESTEP
```

Building a model with Landlab components

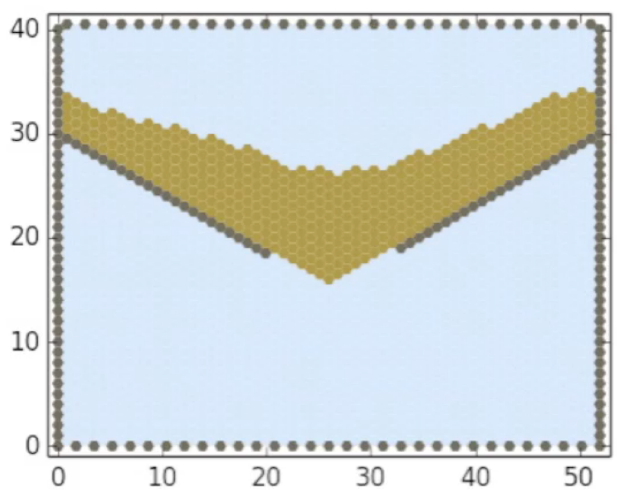


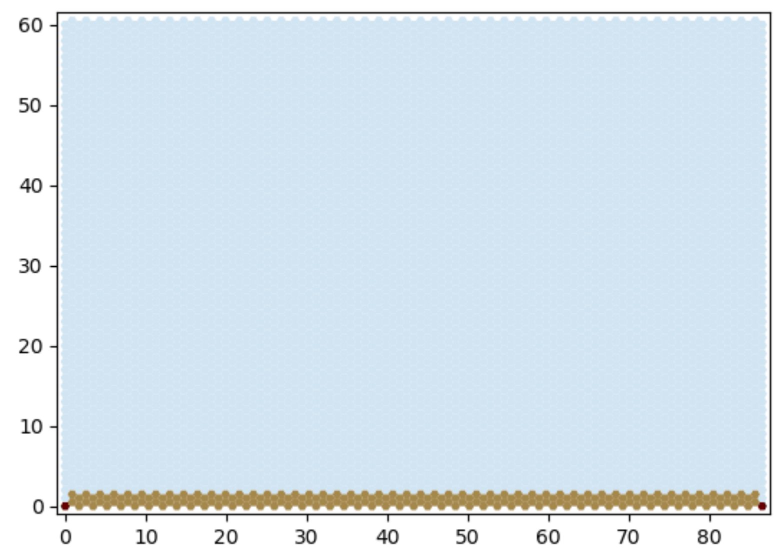
Some process components in the toolkit



Cellular automaton package

CellLab-CTS: Continuous-time stochastic CA



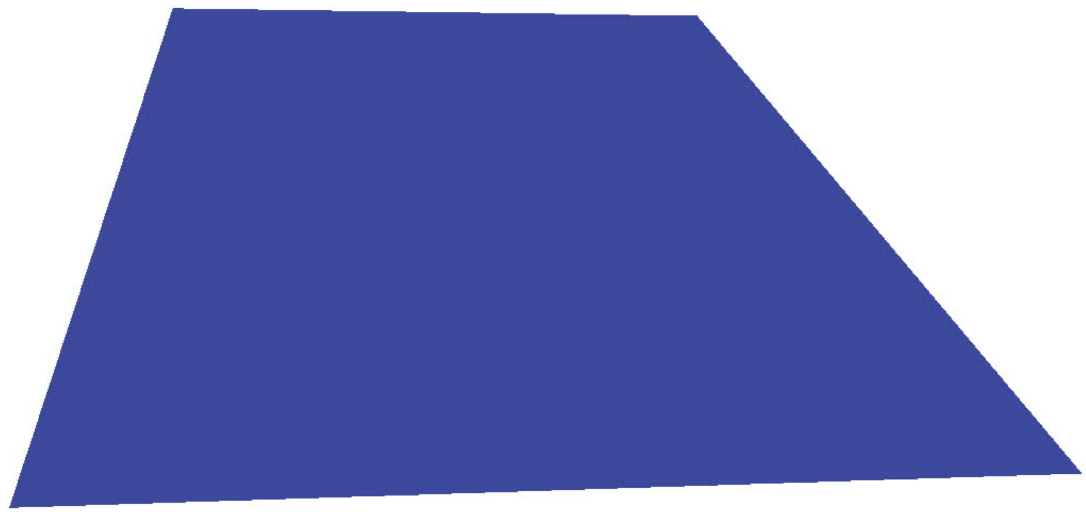


(Tucker et al., 2016 *Geoscientific Model Development;* 2018 *Earth Surface Dynamics*)

Other capabilities

- Input and output
  - Parameter input from formatted text file
  - Read and/or write gridded data:
    - ESRI ASCII `EX: (mygrid, topo) = read_esri_ascii('my_dem.txt')`
    - netCDF
- DEM analysis and pre-processing
  - Configure “watershed” boundary conditions
  - Other terrain analysis functions
- Basic plotting with Matplotlib

Example: 6-line diffusion model



```
for i in range(100):
    g = mg.calc_grad_at_link(z)
    qs[mg.active_links] = -D * g[mg.active_links]
    dqsdx = mg.calc_flux_div_at_node(qs)
    dzdt = uplift_rate - dqsdx
    z[mg.core_nodes] += dzdt[mg.core_nodes] * dt
```

Basic landform evolution model

```
# Create a grid
grid = RasterModelGrid((nrows, ncols), dx)

# Add a field
z = grid.add_zeros('node', 'topographic__elevation')
z += np.random.rand(z.size)/100000.

# Instantiate components
fr = FlowAccumulator(grid, **inputs)
sp = FastscapeEroder(grid, **inputs)
lin_diffuse = LinearDiffuser(grid, **inputs)

# Run loop
for i in range(nt):
    lin_diffuse.run_one_step(dt)
    fr.run_one_step()
    sp.run_one_step(dt)
    z[grid.core_nodes] += uplift_rate * dt
```

Multi-model long-term erosion forecasting

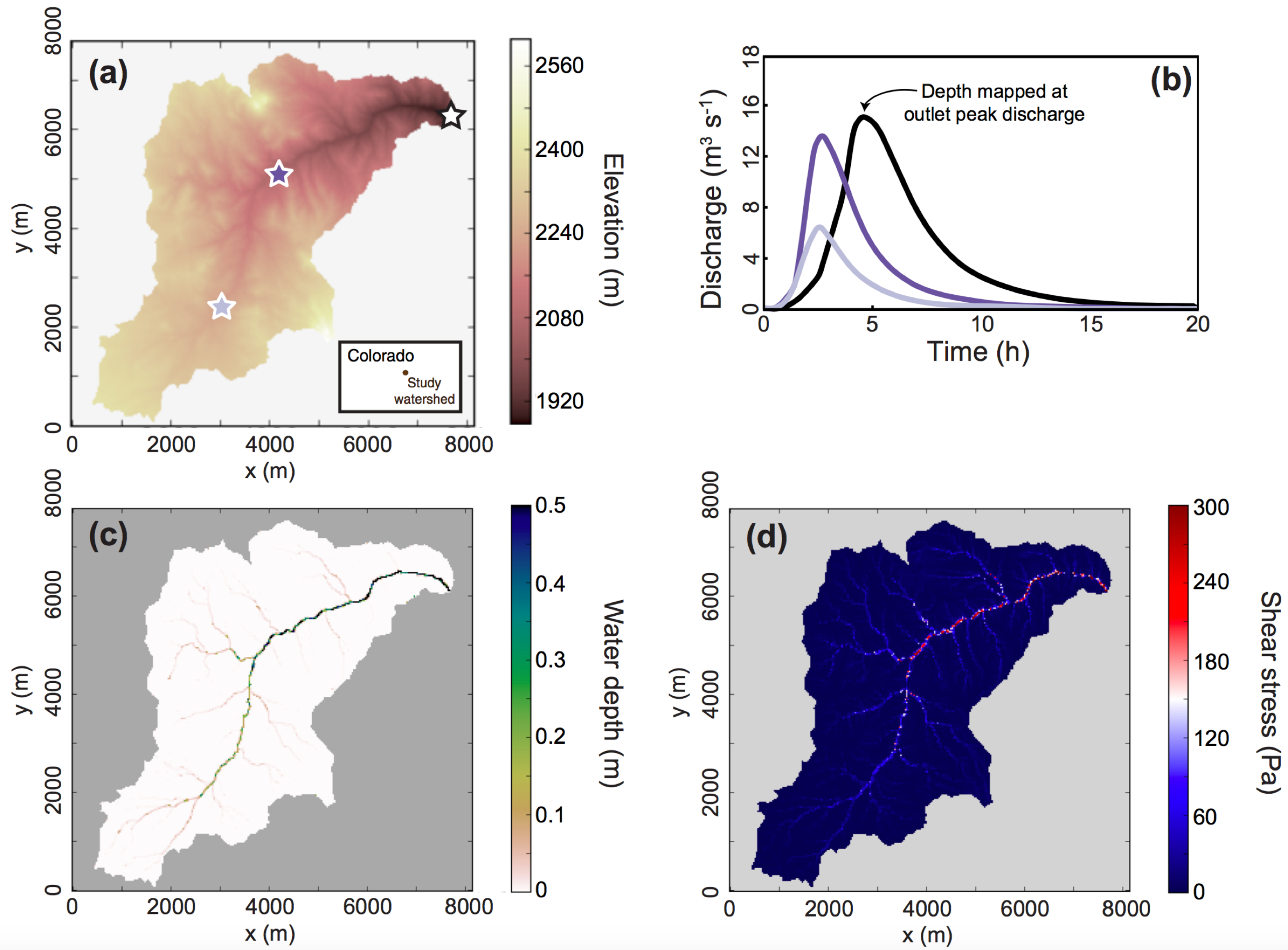
MODEL “BasicRt”

MODEL “BasicChRtTh”

Barnhart et al. (in prep)

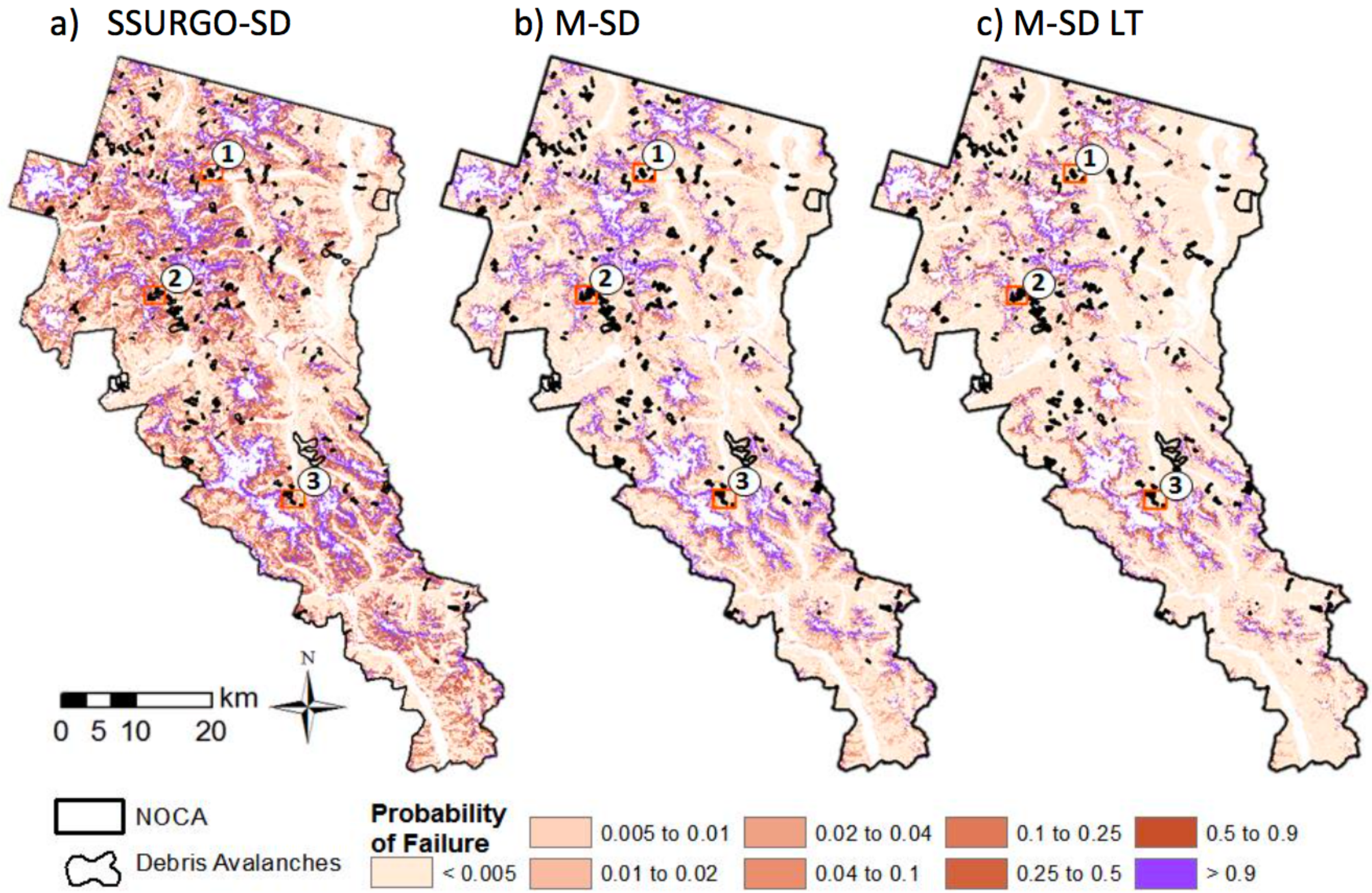
Uses Landlab-built landscape evolution modeling package “TerrainBento” (Barnhart et al., in review)

Basin rainfall-runoff model



(Adams et al., 2017 *Geoscientific Model Development*)

Landslide probability mapping



(Ronda Strauch et al., 2018 *Earth Surface Dynamics*)

Some notebook demos from the Landlab tutorials collection

These and other tutorials can be found via the Landlab website

Tutorial #1: modeling the erosional degradation of a fault scarp using linear diffusion theory

The mathematical problem

SOIL MASS IS CONSERVED ➔

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

η = land-surface elevation

t = time

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

SOIL CREEPS DOWNHILL ➔

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

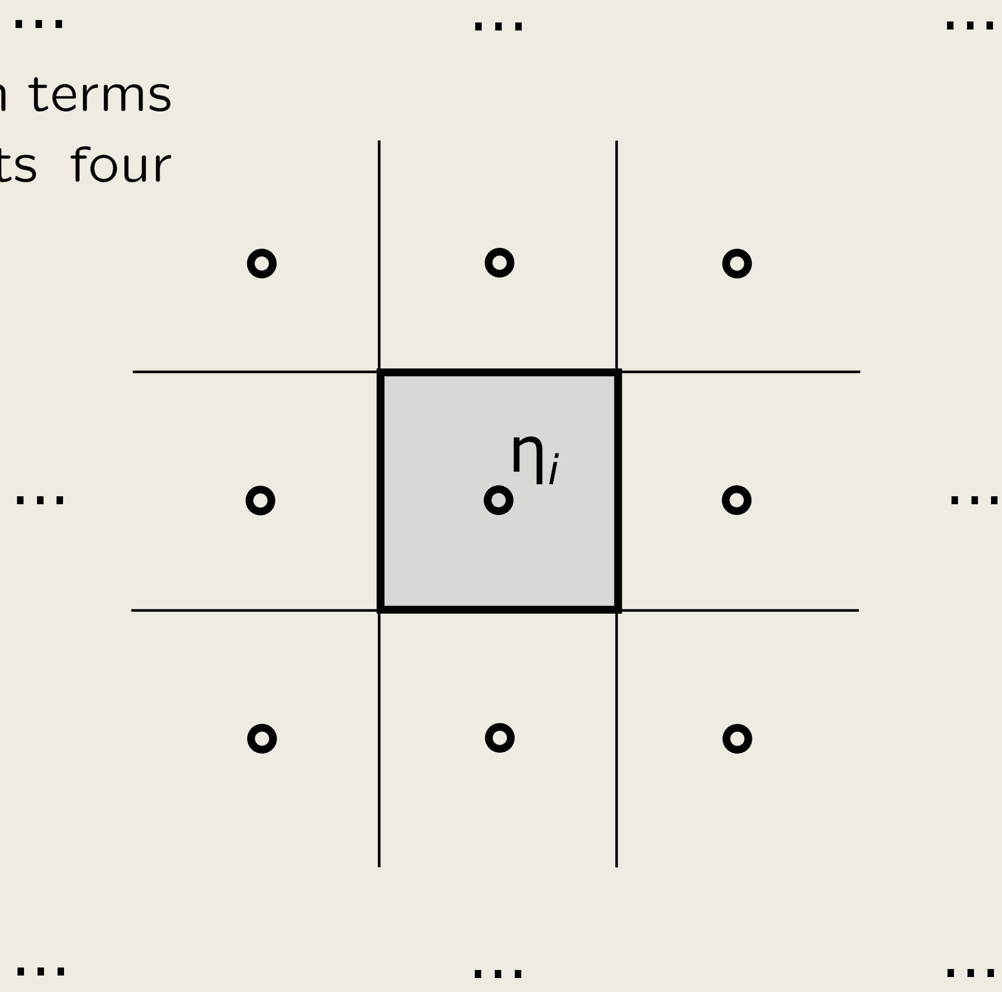
D = transport coefficient $[L^2/T]$

The numerical problem: finite-volume solution scheme

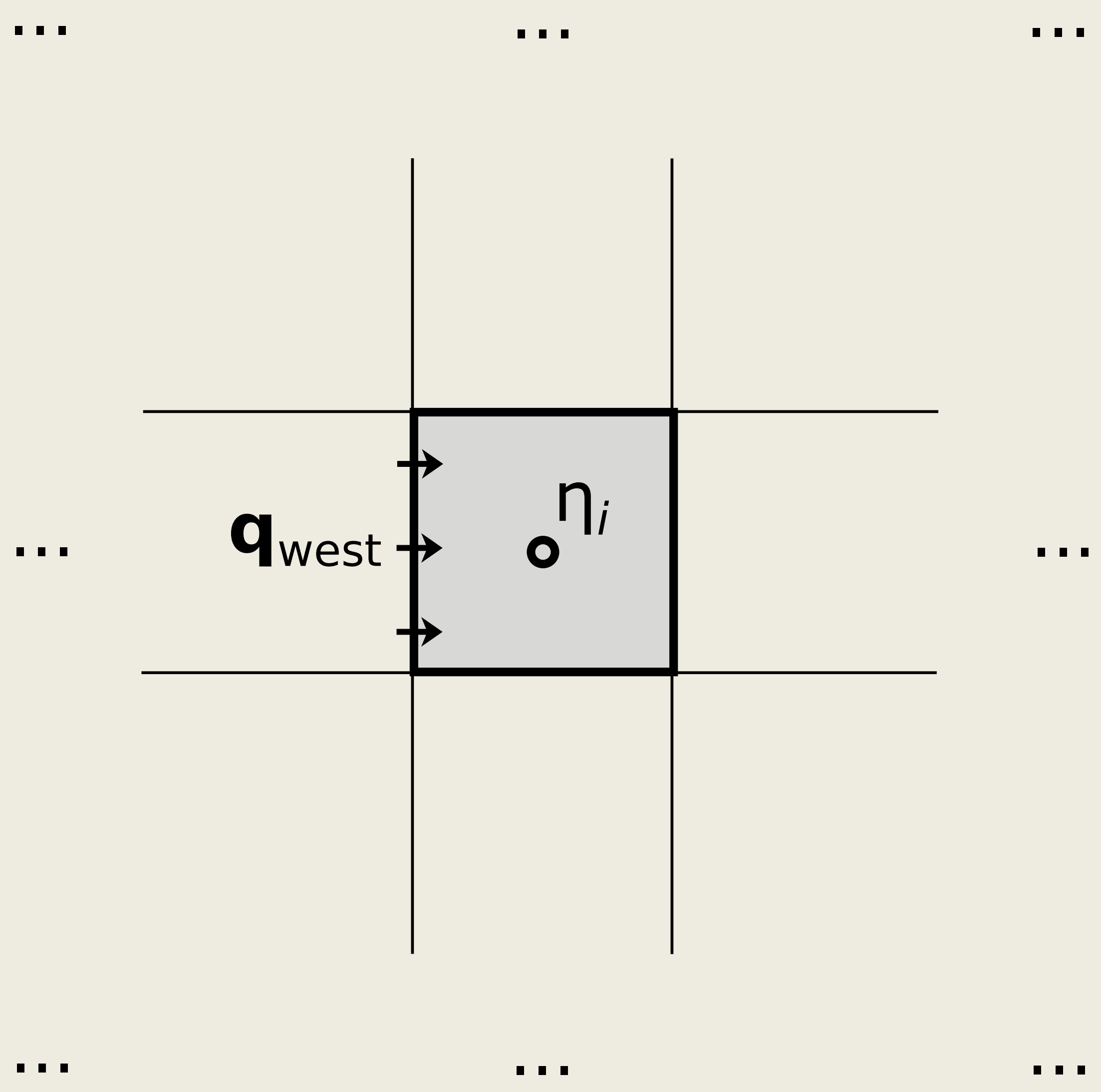
Each interior node i lies within a *cell* whose surface area is A_i.

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

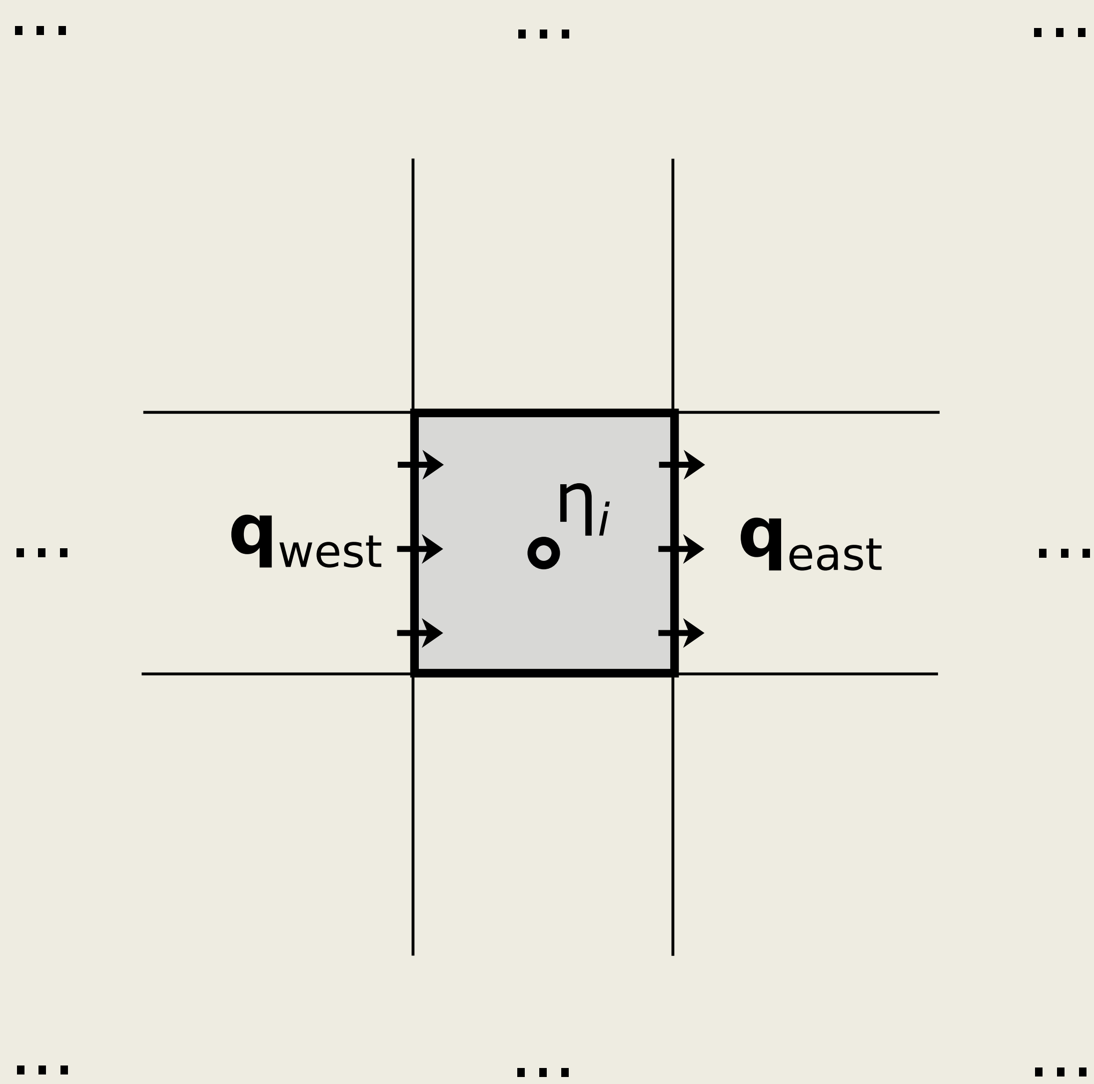
$$\frac{d\eta_i}{dt} = \frac{1}{A_i}\sum_{j=1}^{4} \Delta x q_j$$



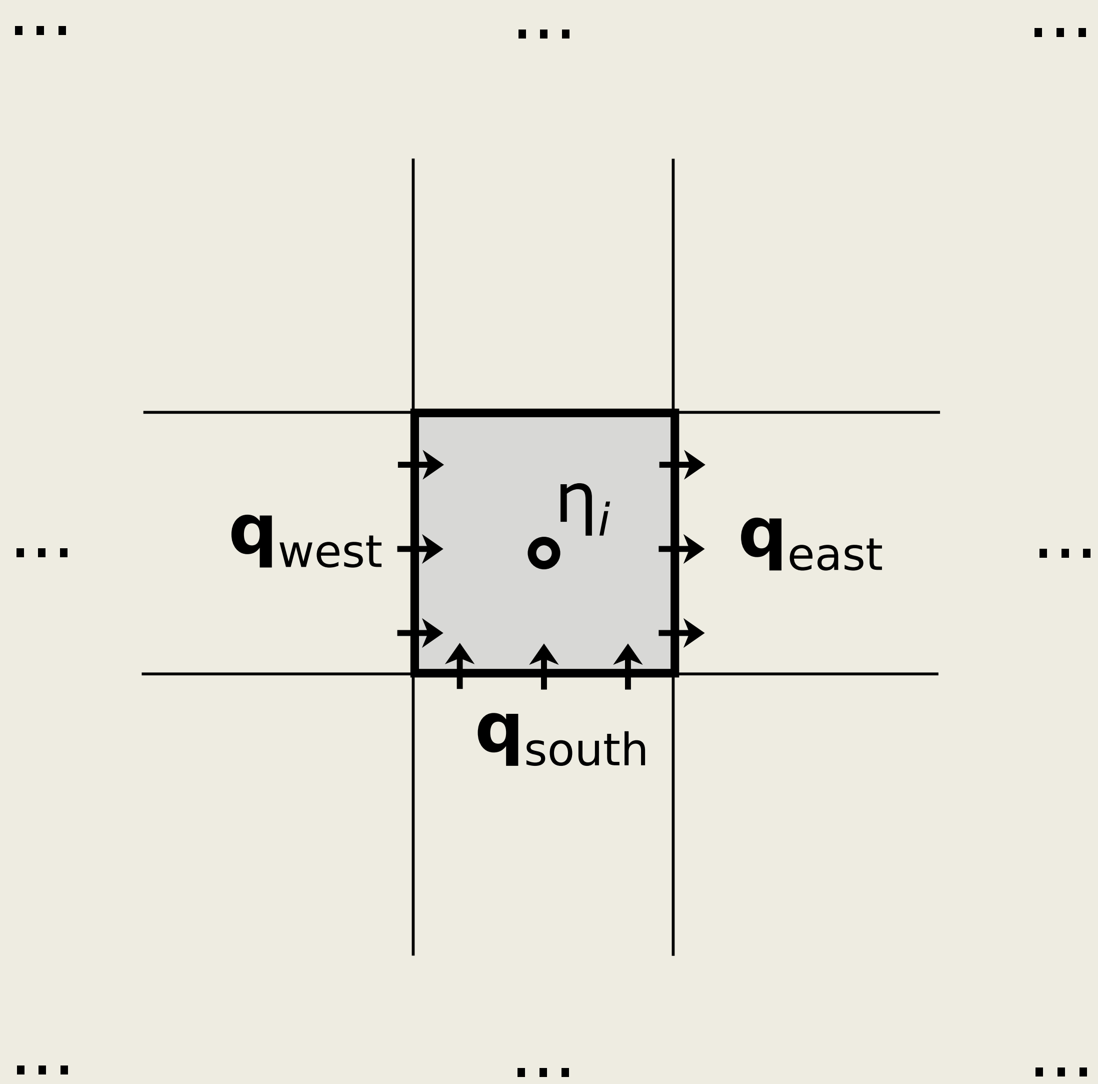
$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\mathrm{west}} \cdots$$



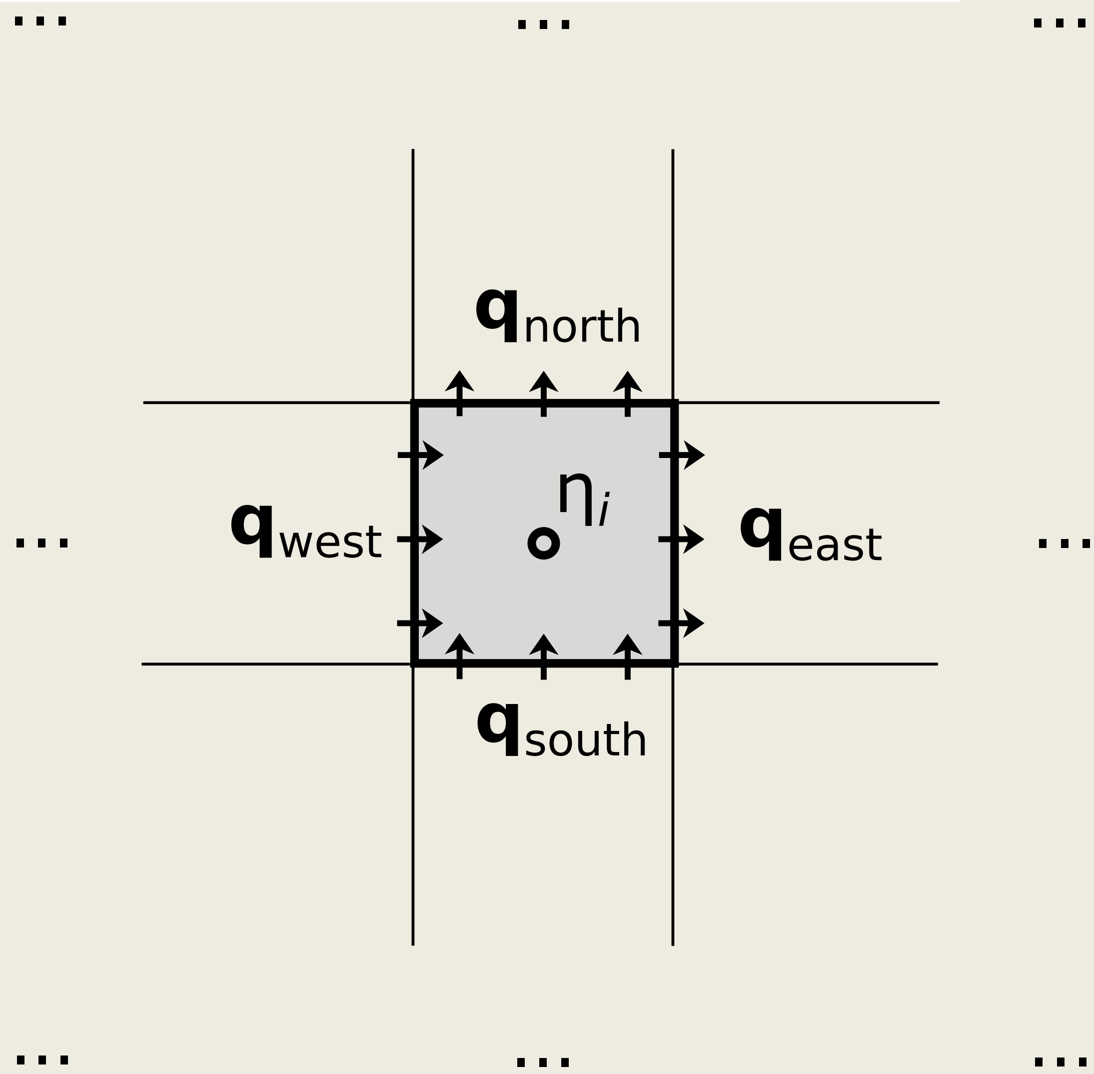
$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\text{west}} - \mathbf{q}_{\text{east}} \cdots$$



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

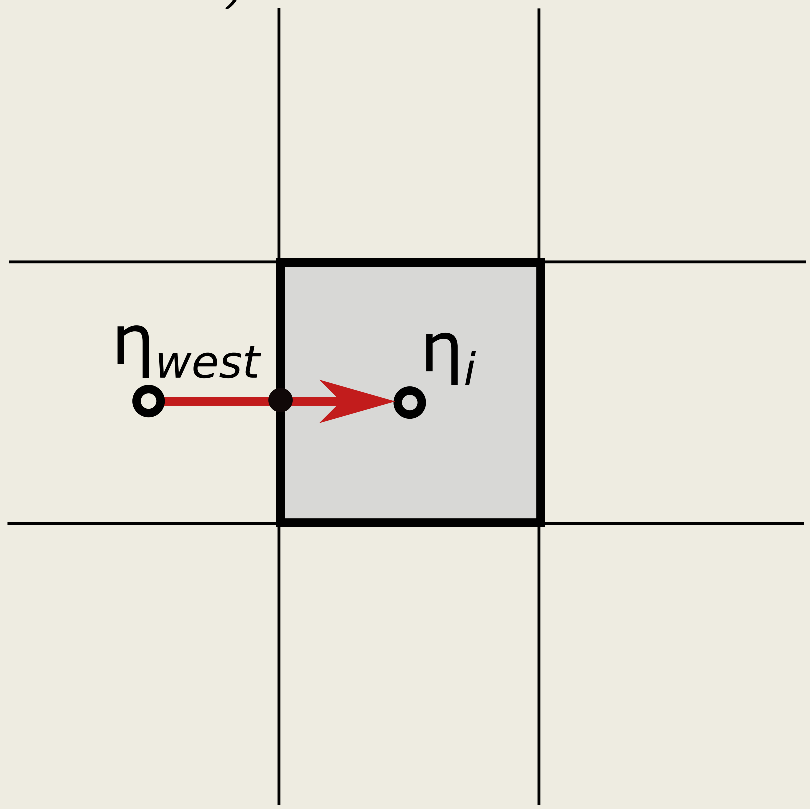


$$\frac{d\eta_i}{dt} = \frac{\Delta x}{A_i}[\mathbf{q}_{\text{west}} - \mathbf{q}_{\text{east}} + \mathbf{q}_{\text{south}} - \mathbf{q}_{\text{north}}]$$



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

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

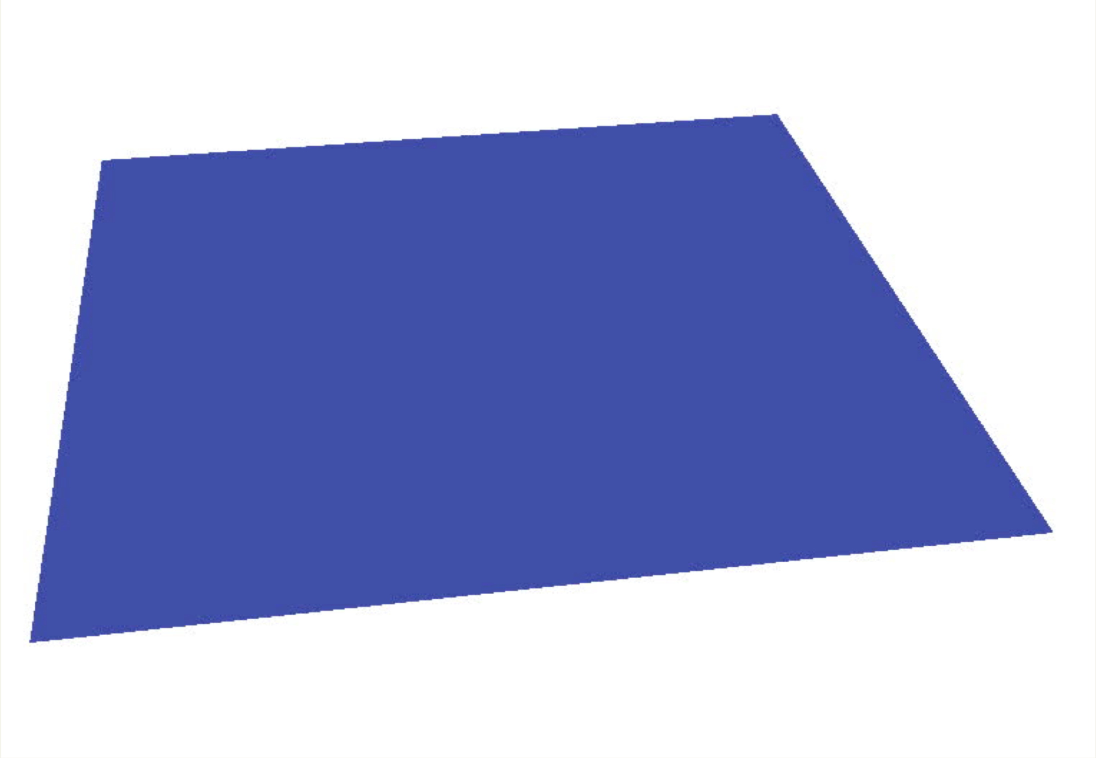


link from η_{west} to η_i

Discretize the time derivative as a forward difference:

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

Tutorial #2: using components to create a basic model of landform evolution



Landlab documentation and other resources

- Website: http://landlab.github.io
- Github site: https://github.com/landlab
- Jupyter notebook tutorials
- Online user guide
- API reference manual
- Best way to pose a questions or make a request: post an ISSUE on Github site
- Site visits

Questions?