Introduction to Landlab

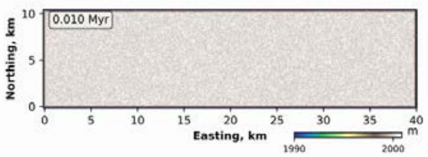
Getting to know the Grid

and Working with Components

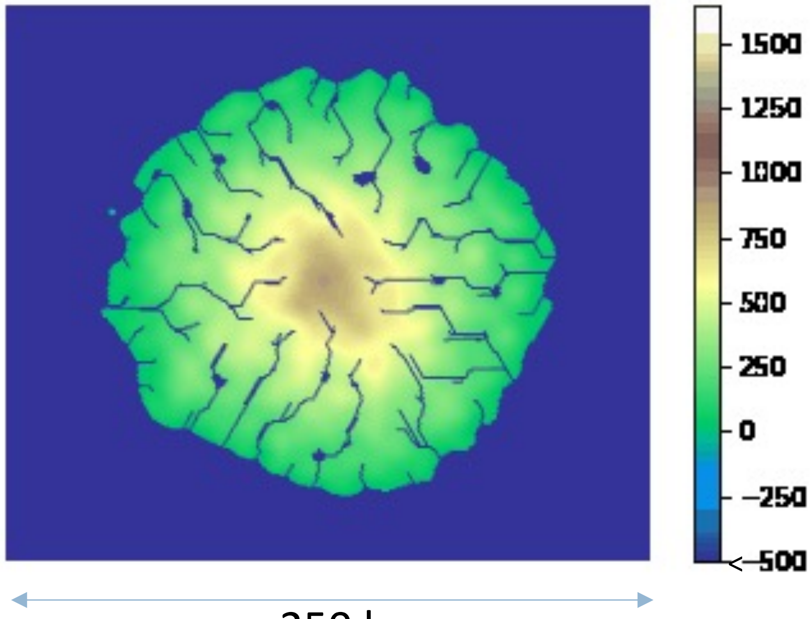
Greg Tucker, CU Boulder
overview notes for "Introduction to Landlab" clinic
CSDMS Annual Meeting, May 2022

Landlab Toolkit

a Python package for building integrated, modular numerical models of diverse surface processes



(animation by B. Campforts)



250 km

Create 2D **grids** as data objects

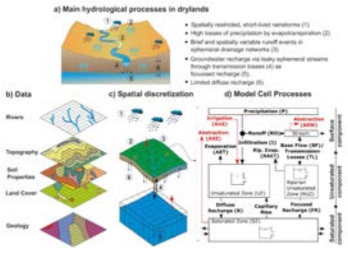
Populate a grid with **fields** of data

Create integrated models from reusable **components**

Geared toward, but not limited to, surface processes

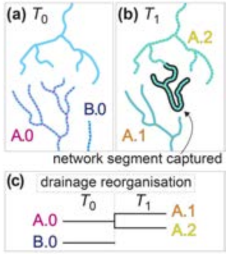
One element in CSDMS' Open**Earthscape** modeling system

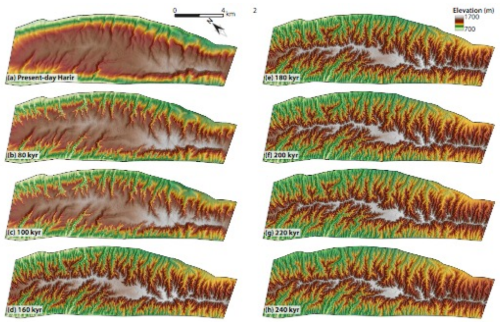
Examples of recent studies using Landlab



Dryland hydrology (Quichimbo et al., 2021)

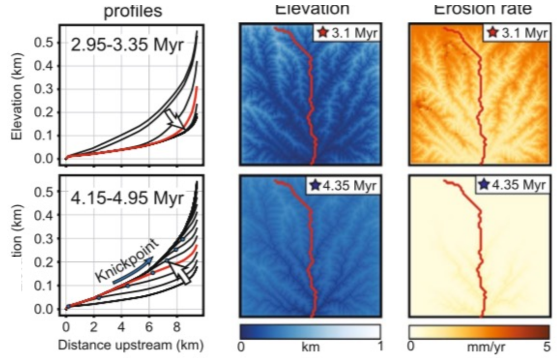
Species diversity & river capture (Lyons et al., 2022)

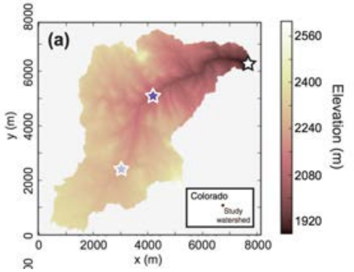


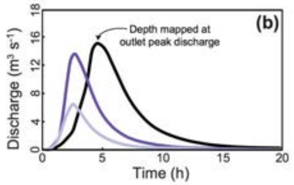


Evolution of anticlines (Zebari et al., 2019)

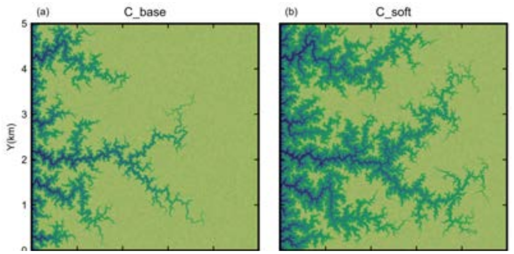
Sediment provenance as a signal of climate and tectonics in sedimentary basins (Sharman et al., 2019)



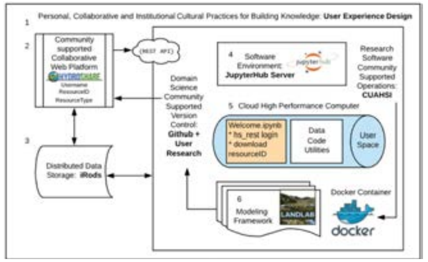




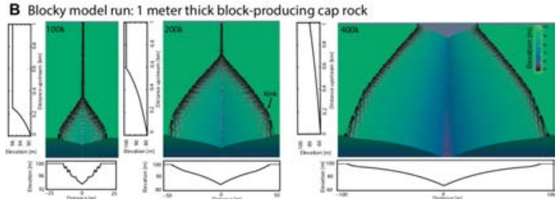
Runoff hydrodynamics (Adams et al., 2017)



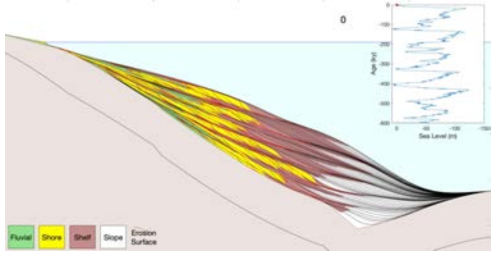
Post-glacial drainage nets (Lai & Anders, 2017)



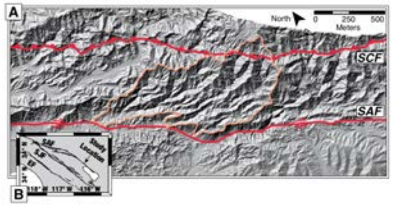
Hydrology education (Bandaragoda et al., 2019)



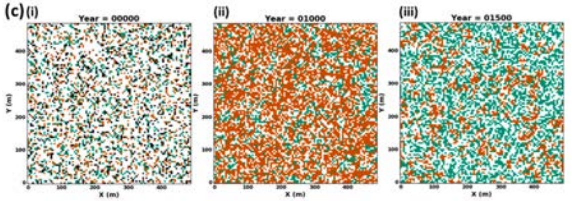
Influence of boulders on hillslope & channel evolution (Glade, Shobe et al., 2019)



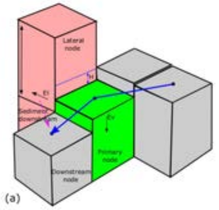
Basin stratigraphy (Steckler et al., in prep)



Tectonic shear (Gray et al., 2017)

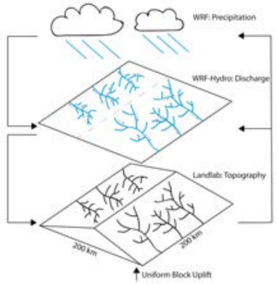


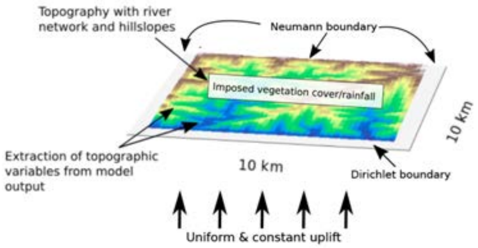
Vegetation dynamics (Nudurupati et al., in review)



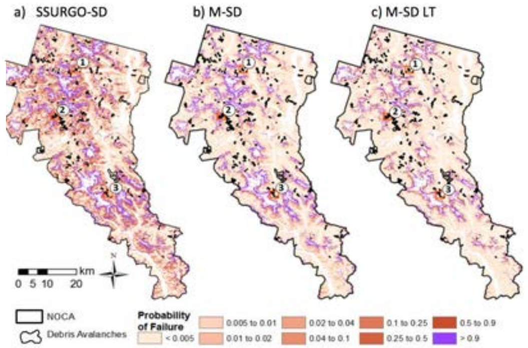
Valley widening (Langston et al., 2018)

Climate-landscape with WRF (Shen et al., 2021)

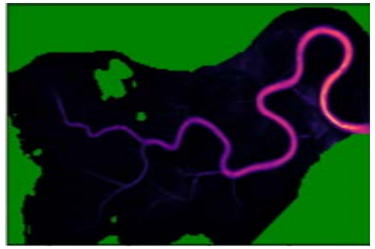




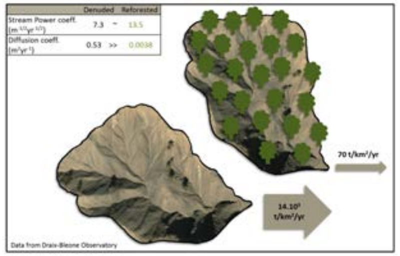
Vegetation & erosion (Schmid et al., 2018)



Landslide susceptibility (Strauch et al., 2018)

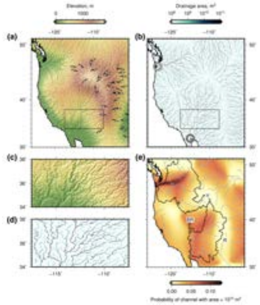


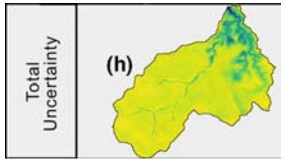
Flow in tidal channels



Sediment yield (Carriere et al., 2019)

Mantle & river patterns (Lipp & Roberts, 2021)

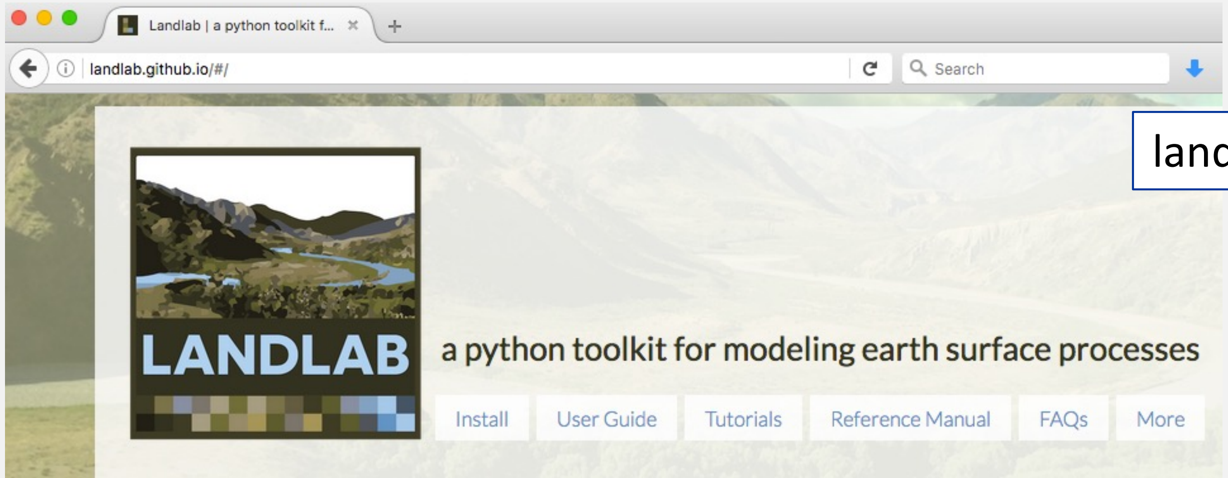




Erosion projection (Barnhart et al., 2020)

Resources for learning about Landlab

- Overview papers:
  - Hobley et al. (2017) “all about”: https://doi.org/10.5194/esurf-5-21-2017
  - Barnhart et al. (2020) Landlab 2.0: https://doi.org/10.5194/esurf-8-379-2020
  - Tucker et al. (2022) CSDMS: https://doi.org/10.5194/gmd-15-1413-2022
  - Digital repository on GitHub: https://github.com/landlab
- Website, documentation, and tutorials: https://landlab.github.io



landlab.github.io

https://landlab.readthedocs.io/en/latest/user_guide/tutorials.html

LANDLAB

An open-source Python package for building numerical models of Earth surface dynamics.

Watch 27

Quick search

Go

Navigation

- Installation Instructions
- Getting Started
- User Guide
  - Introduction to Python
  - The Landlab Grid
  - Model with Landlab and Components
  - Landlab and Units
  - Landlab Tutorial Library
  - Additional resources
  - Presentations, Clinics, and Classroom Use
  - Overland flow User Guide
  - CellLab-CTS User Guide
  - Major Version Transition Guides

Tutorials

The Landlab Tutorials provide examples of Landlab core concepts and component introductions. Tutorials exist as interactive Jupyter notebooks that contain alternating cells of computer code and text that explain the code. In addition to Landlab Tutorials that exemplify Landlab, notebooks intended to teach and learn surface dynamics are the Landlab Teaching Tutorials.

Launch notebooks online

Landlab Notebooks can be accessed online with the following link: Binder. Here the notebooks are provided within a binder online environment that includes Landlab.

The welcome page on Binder provides onward links to most of our tutorials. If you're a newbie you might want to skip directly to a recommended syllabus for learning Landlab here.

Launch notebooks locally

How you installed Landlab determines the steps to launch notebooks that use a copy of Landlab on your computer.

User installations

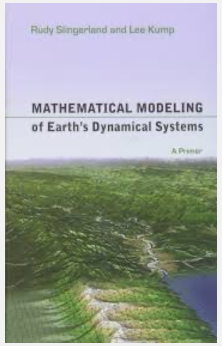
If you installed using a prepackaged binary, the most common method, the notebooks can be run in a conda environment. These instructions describe how to create a conda environment for the notebooks.

Once the conda environment has been created, it must be activated and then the notebooks can be launched:

```
$ conda activate landlab_notebooks
$ jupyter notebook notebooks/welcome.ipynb
```

Landlab prerequisites:

- Python programming, including use of classes and objects
- Numpy arrays
- Relevant numerical methods

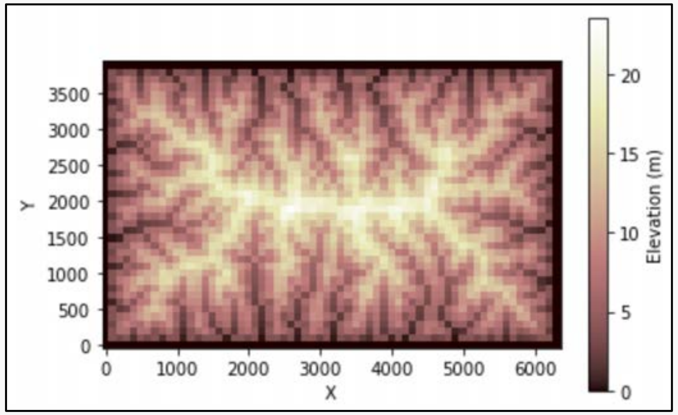


Slingerland & Kump (2011) Mathematical Modeling of Earth's Dynamical Systems

Press et al. (2007) Numerical Recipes: The Art of Scientific Computing

```python
# imports
import numpy as np
from landlab import RasterModelGrid, imshow_grid
from landlab.components import FlowAccumulator, StreamPowerEroder, LinearDiffuser

# make a grid
grid = RasterModelGrid((40, 64), xy_spacing=100.0)

# make a field and initialize it
topo = grid.add_zeros('topographic__elevation', at='node')
topo[grid.core_nodes] += np.random.rand(len(grid.core_nodes))

# instantiate components
fa = FlowAccumulator(grid, flow_director='D8')
sp = StreamPowerEroder(grid, K_sp=0.0001)
ld = LinearDiffuser(grid, linear_diffusivity=0.01)

# run model
for i in range(1000):
    topo[grid.core_nodes] += 0.1  # uplift
    ld.run_one_step(250.0)        # soil creep / hillslope processes
    fa.run_one_step()             # route flow
    sp.run_one_step(250.0)        # water erosion

# plot
imshow_grid(grid, topo, colorbar_label='Elevation (m)')
```

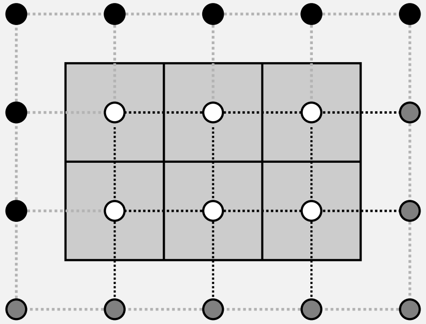
A landscape evolution model in 15 lines



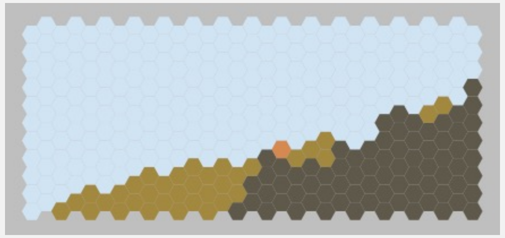
Grids

- Create a grid in one line of code
- Choose among different grid types
- Each grid is composed of graph primitives such as **nodes** and **links**
- Grid objects include all data to describe grid topology and geometry

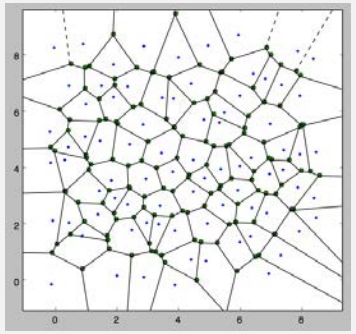
RASTER



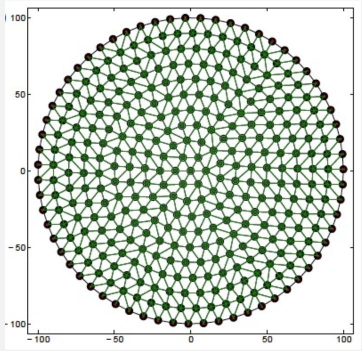
HEXAGONAL



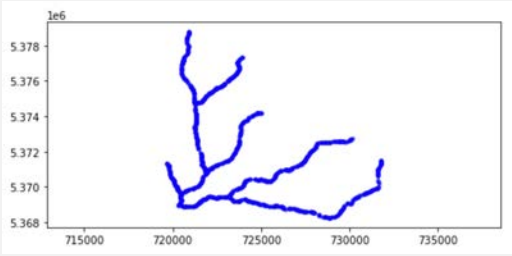
VORONOI / DELAUNAY



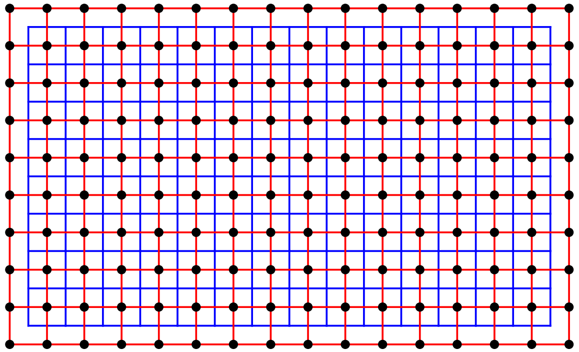
RADIAL

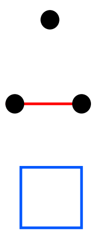


NETWORK

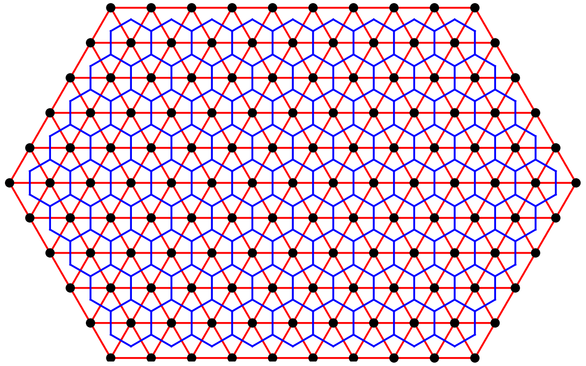


```
grid = RasterModelGrid(shape=(10, 16))
```

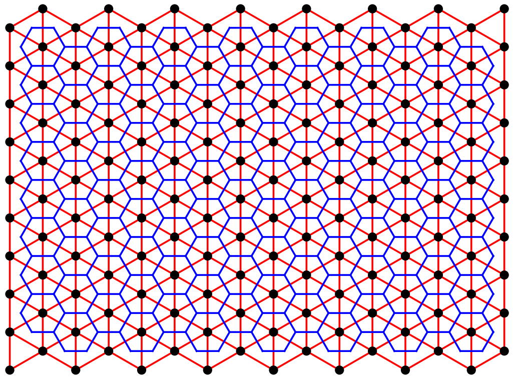


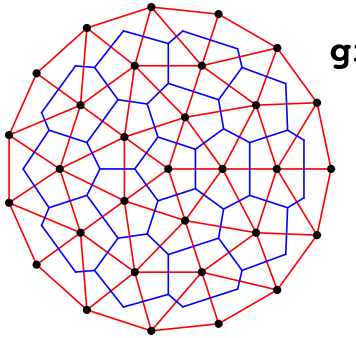



```
grid = HexModelGrid(shape=(11, 10))
```



```
grid = HexModelGrid(
    shape=(16, 10),
    node_layout='rect',
    orientation='vertical')
```





```
grid = RadialModelGrid((3, 5))
```

🡐 Delaunay triangulation with Voronoi polygons

Fields

- A **field** is a flat array of data associated with a particular type of grid element
- Example:

```
elev = grid.add_zeros('topographic__elevation', at='node')
```

MAKE A NEW FIELD FILLED WITH ZEROS

GIVE IT A NAME

ONE VALUE PER GRID NODE

```
elev is grid.at_node['topographic__elevation']
    True
```

ACCESS IT BY NAME VIA THE GRID

See **User Guide** section on **Adding Data to a Landlab Grid Element using Fields:**
https://landlab.readthedocs.io/en/latest/user_guide/grid.html#adding-data-to-a-landlab-grid-element-using-fields

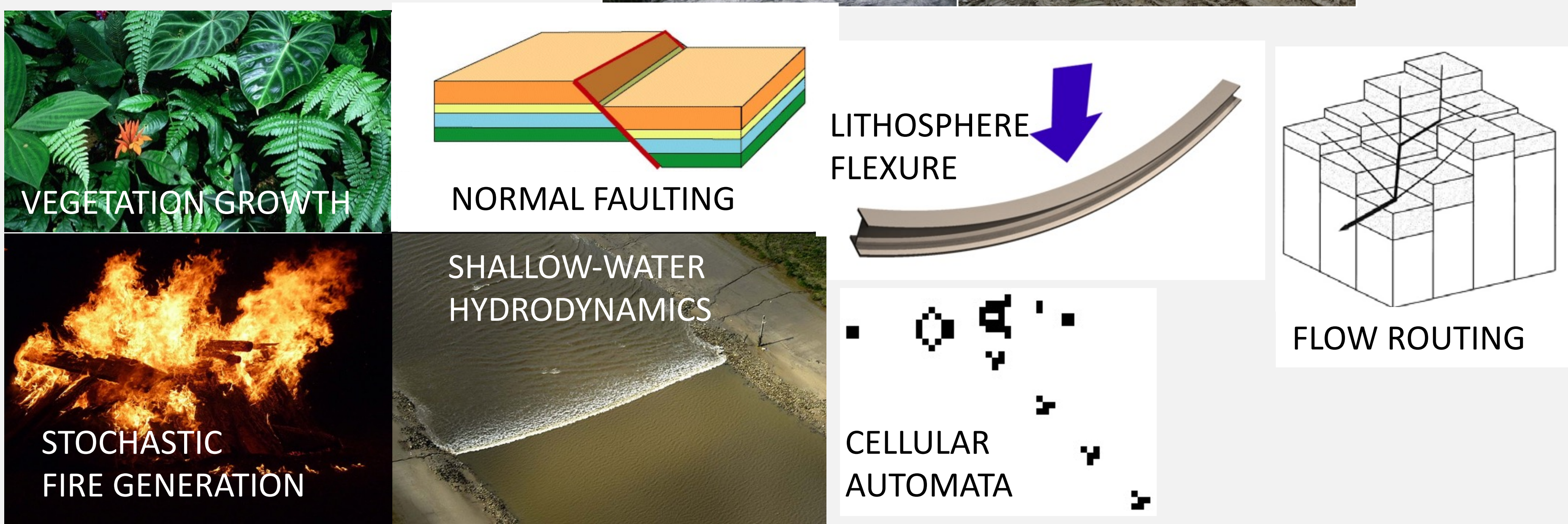
And the **Working with Fields** tutorial

Landlab components

A **component** is a Python class with a semi-standard interface that implements:

- a model for a particular process, or
- a calculation for one kind of analysis

Landlab 2.4 includes about 70 components



https://landlab.readthedocs.io/en/latest/reference/components/index.html

LANDLAB

An open-source Python package for building numerical models of Earth surface dynamics.

Watch 27

Quick search

Go

Navigation

Installation Instructions
Getting Started
User Guide
API reference
- Basic Model Interface
- Cellular Automata (CA)
- Command Interface
- Components
- Core Landlab Classes
- Data Record
- Landlab Fields
- Landlab Framework
- Landlab Graph
- Landlab Grids
- Input/Output (IO)
- Layers
- Plotting and Visualization
- Utilities and Decorators
- Values
- Full Index
- References

Guide for Developers
Release Notes

landlab @ GitHub
Contact Us

Powered by CSDMS

Components

This section contains documentation and API reference information for the following categories of components:

Hillslope geomorphology

- LinearDiffuser: Model soil creep using "linear diffusion" transport law (no depth dependence)
- PerronNLDiffuse: Model soil creep using implicit solution to nonlinear diffusion law
- DepthDependentDiffuser: depth dependent diffusion after Johnstone and Hilley (2014)
- TransportLengthHillslopeDiffuser: Hillslope diffusion component in the style of Carretier et al. (2016), and Davy and Lague (2009)
- TaylorNonLinearDiffuser: Model non-linear soil creep after Ganti et al. (2013)
- DepthDependentTaylorDiffuser: Model depth dependent non-linear soil creep after Ganti et al. (2012) and Johnstone and Hilley (2014)

Fluvial geomorphology

- FastscapeEroder: Compute fluvial erosion using stream power theory ("fastscape" algorithm)
- StreamPower: Compute fluvial erosion using stream power theory (also uses "fastscape" algorithm but provides slightly different formulation and options)
- SedDepEroder: Compute fluvial erosion using using "tools and cover" theory
- StreamPowerSmoothThresholdEroder: Compute fluvial erosion using stream power theory with a numerically smoothed threshold
- DetachmentLtdErosion: Solve stream power equations, but without stability checks
- ErosionDeposition: Fluvial erosion in the style of Davy and Lague (2009)
- Stream Power with Alluvium Conservation and Entrainment
  - Space: Stream Power with Alluvium Conservation and Entrainment
  - SpaceLargeScaleEroder: SPACE large-scale eroder
- NetworkSedimentTransporter: Lagrangian sediment transport in a river network

Flow routing

- The Landlab FlowDirectors: Components for Flow Direction
  - FlowDirectorSteepest
  - FlowDirectorD8
  - FlowDirectorMFD
  - FlowDirectorDinf
- FlowAccumulator: Component to do FlowAccumulation with the FlowDirectors
- LossyFlowAccumulator: Component to accumulate flow with the FlowDirectors, while water is lost or gained downstream
- Functions to support flow accumulation
  - Route-to-one methods
  - Route-to-multiple methods
- DepressionFinderAndRouter: Handle depressions in terrain by calculating extent and drainage of "lakes"
- LakeMapperBarnes: Component to temporarily fill depressions and reroute flow across them
- PriorityFloodFlowRouter: Accumulate flow and calculate drainage area using RICHDEM
- SinkFiller: Permanently fill pits in a topography

Shallow water hydrodynamics

- OverlandFlow: Model shallow water flow over topography using the numerical approximation of de Almeida
- OverlandFlowBates: Model shallow water flow over topography using the numerical approximation of Bates
- KinematicWaveRengers
- KinwaveImplicitOverlandFlow
- KinwaveOverlandFlowModel
- LinearDiffusionOverlandFlowRouter
- TidalFlowCalculator: Calculate cycle-averaged tidal flow velocity using method of Mariotti (2018)

Land surface hydrology

- Radiation: Calculate solar radiation on topography given latitude, date, and time
- PotentialEvapotranspiration: Compute potential evapotranspiration
- SoilMoisture: Compute the decay of soil moisture saturation at storm-interstorm time period
- SoilInfiltrationGreenAmpt: Model infiltration of surface water according to the Green-Ampt equation

Groundwater hydrology

- GroundwaterDupuitPercolator: model flow in a shallow unconfined aquifer using the Dupuit-Forcheimer approximation

Landslides

- BedrockLandslider: Location and magnitude of episodic bedrock landsliding
- Landslides: Compute probability of failure for shallow landslides
- DimensionlessDischarge: Testing thresholds of debris flows in stream segments following Tang et al.

Vegetation

- Vegetation: Model plant dynamics using multiple representative plant species
- VegCA: Simulate plant competition with cellular automaton model for grass, shrubs, and trees

Biota

- Species Evolution
  - SpeciesEvolver: Component to evolve life in a landscape
  - ZoneController: Control zones and populates them with taxa
  - ZoneTaxon: A zone-based taxon

Precipitation

- PrecipitationDistribution: Generate random sequence of precipitation events
- SpatialPrecipitationDistribution: Generate random sequence of spatially-resolved precipitation events

Weathering

- ExponentialWeatherer: exponential soil production function in the style of Ahnert (1976)
- ExponentialWeathererIntegrated: exponential soil production function in the style of Ahnert (1976) integrated in dt

Subaqueous / Submarine Processes

- CarbonateProducer: Grow carbonate strata using growth function of Bosscher and Schlager (1992)
- SimpleSubmarineDiffuser: Calculate underwater sediment transport using water-depth-dependent diffusion

Tectonics

- ListricKinematicExtender: Simulate Extensional Tectonic Motion on a Listric Fault Plane

Terrain Analysis

- SteepnessFinder: Calcuate steepness and concavity indices from gridded topography
- ChiFinder: Perform chi-index analysis for gridded topography
- DrainageDensity: Calculate drainage density from topography
- Profiler: Create and plot profiles
- ChannelProfiler: Create and plot channel profiles
- TrickleDownProfiler: Create and plot trickle down profiles
- HackCalculator: Calculate Hack's law coefficients
- HeightAboveDrainageCalculator: Calculate height above nearest drainage

Tectonics

- Flexure: Calculate elastic lithosphere flexure under given loads (assumes uniform flexural rigidity)
- Functions used to calculate lithospheric deflection
- landlab.components.flexure.ext package
  - Submodules
  - landlab.components.flexure.ext.flexure1d module
  - Module contents
- gFlex: Compute elastic lithosphere flexure with variable rigidity
- NormalFault: Vertical uplift of grid nodes based on a user-specified uplift time series

Fire

- FireGenerator: Generate random sequence of fire events

Fracture Generation

- FractureGridGenerator: Generate random fracture patterns on a regular raster grid

Lithology

Two objects based on the EventLayers object exist to make it easier to deal with spatially variable lithology and associated properties. The Lithology components contain information about spatially variable lithology and connect with the Landlab model grid so that when rock is eroded or advected upward by rock uplift the values of rock propeties at the topographic surface are updated.

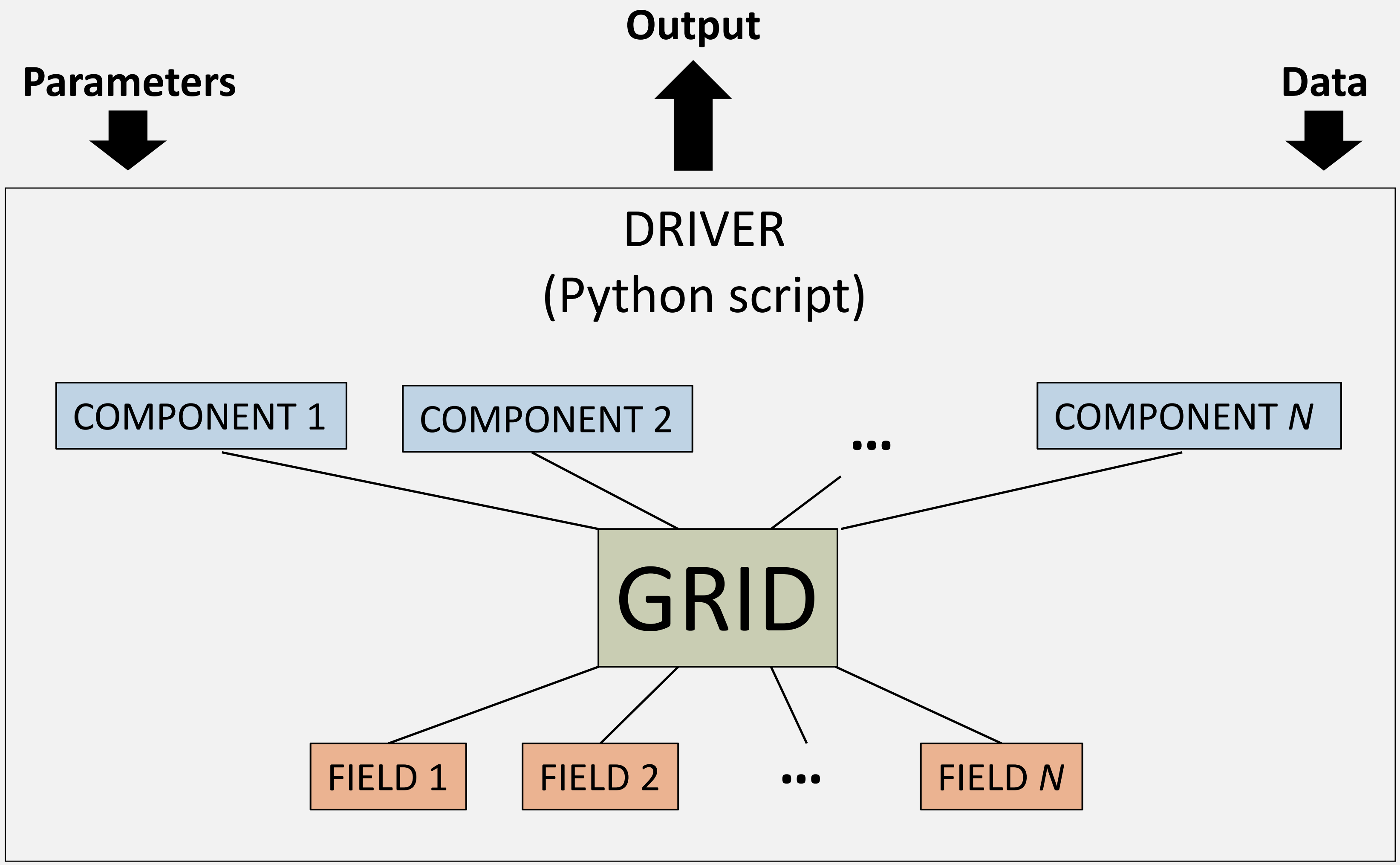
First is the Lithology component which is a generic object for variable lithology.

- Lithology: Create a 3D representation of variable lithology

Second is LithoLayers which makes it easy to make layered rock.

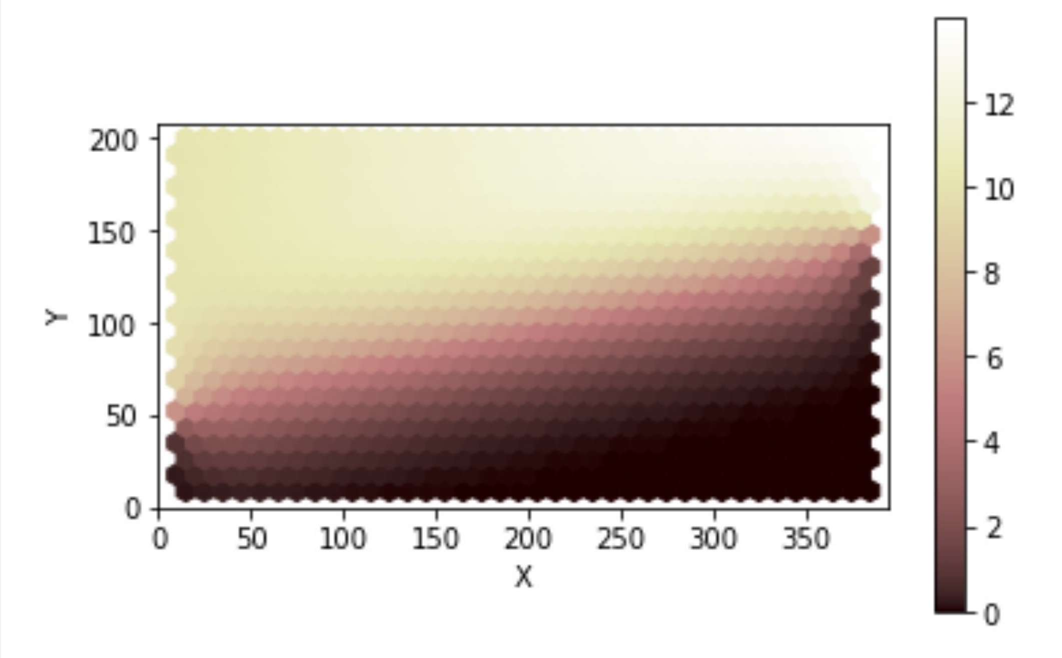
- LithoLayers: Create a layered lithology

Building a model with Landlab **components**



Numerical functions

- Landlab's graph-based data structures support different types of numerical method
- Some numerical functions are built in; other yet to be created…




```
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
```

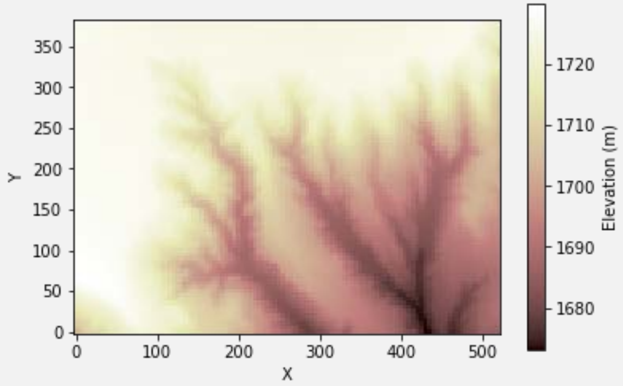
See tutorials on: landlab_fault_scarp, gradient_and_divergence, matrix_creation

Data I/O and related

- Input and output
  - Parameter input from formatted text file
  - Read and/or write gridded data:
    - ESRI ASCII
    - netCDF
    - .obj

EX: `(mygrid, topo) = read_esri_ascii('my_dem.txt')`

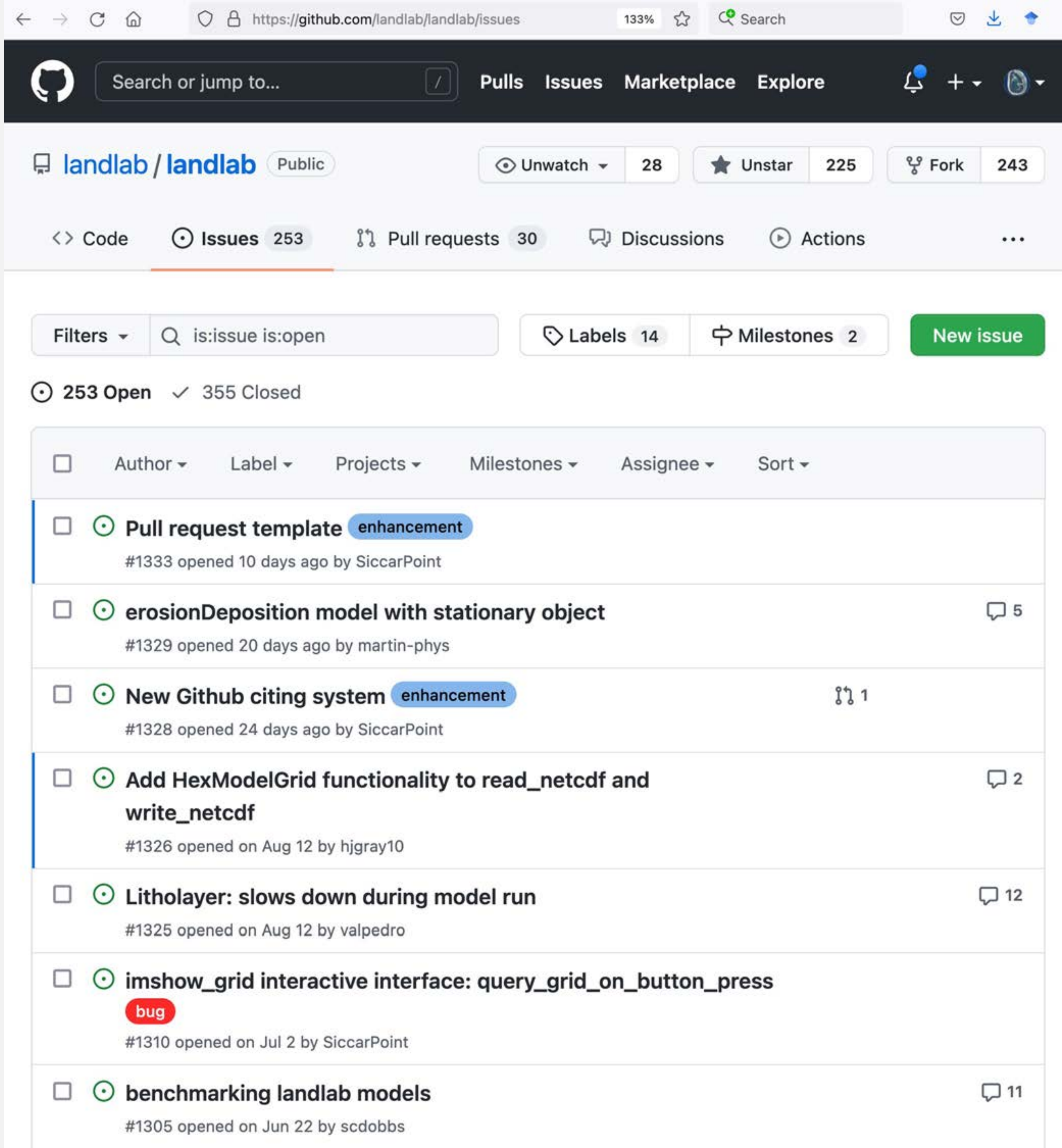
- DEM analysis and pre-processing
  - Configure "watershed" boundary conditions
  - Other terrain analysis functions



RESOURCES: Refence manual section on **Input/Output**
Tutorials on **reading_dem_into_landlab** & various **terrain analysis** tools

Contributing and getting involved!

- Landlab was made by and for the scientific community
- Lots of ways to contribute:
  - Raising or answering issues
  - Reviewing pull requests
  - New components and capabilities



Questions?

gtucker@colorado.edu

csdms@colorado.edu

https://landlab.github.io