ANUGA's underlying Algorithm

Stephen Roberts[1]

[1]Department of Mathematics
The Australian National University

September 15, 2008

Outline

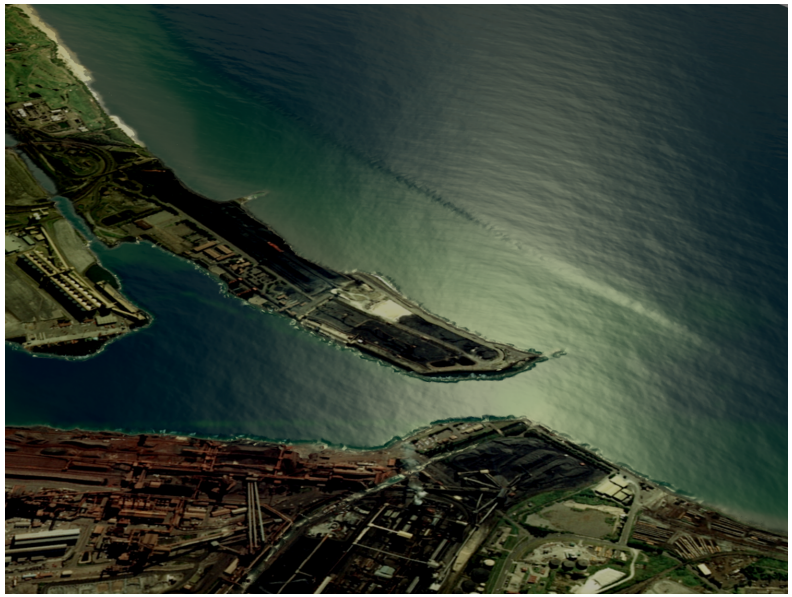
1. ANUGA Inundation Model
2. Algorithm

Some Members of the Development Team

- GeoSciences Australia: Ole Nielsen, Duncan Gray, Adrian Hitchman, Chris Zoppou, Trevor Dhu
- ANU: Stephen Roberts, Darran Edmundson, Drew Whitehouse, Matthew Hardy, Linda Stals, Peter Row, Jack Kelly, John Jakeman

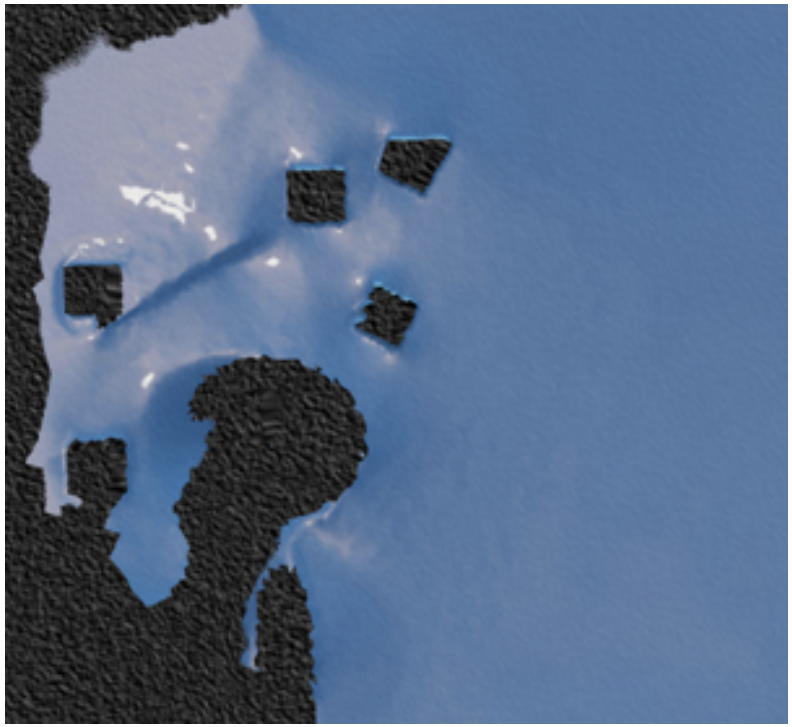
AnuGA Inundation Model

- Water flow described by Shallow Water Wave Equation
- Discretised using finite-volumes method (triangular mesh)
- Object Oriented implementation
- Able to handle realistically sized situations
- Suitable for a wide range of flow problems
- Based exclusively on Open Source Software Components



Finite Volume Method

- Fluxes computed using the Central Scheme
- Time stepping based on weighted combinations of First order Euler.
- Time step determined by triangle size and wave speed (CFL)
- Second order with respect to spatial dimensions



Quantities and Equations

- Conserved quantities:
  - Water stage ($w = z + h$)
  - (or) Water Depth (h)
  - Horizontal momentum (uh, vh)
- Other quantities:
  - Bed elevation (z)
  - Friction (η)
  - Other
- Shallow Water Wave Equation

$$\begin{bmatrix} h \\ uh \\ uv \end{bmatrix}_t + \begin{bmatrix} uh \\ uuh + \frac{1}{2}gh^2 \\ uvh \end{bmatrix}_x + \begin{bmatrix} vh \\ vuh \\ vvh + \frac{1}{2}gh^2 \end{bmatrix}_y = \begin{bmatrix} 0 \\ -ghz_x \\ -ghz_y \end{bmatrix}$$

Form of Equations

Shallow Water Wave Equations has form

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial G}{\partial y} = S$$

Integrate over each cell (and apply integration by parts) to obtain

$$\frac{dU_i}{dt} + \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} H_{ij} l_{ij} = S_i$$

- U_i the vector of conserved quantities averaged over the ith cell,
- S_i is the source term associated with the ith cell, and
- H_{ij} is the outward normal flux of material across the ijth edge.

Form of Equations

Shallow Water Wave Equations has form

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial G}{\partial y} = S$$

Integrate over each cell (and apply integration by parts) to obtain

$$\frac{dU_i}{dt} + \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} H_{ij} l_{ij} = S_i$$

- U_i the vector of conserved quantities averaged over the ith cell,
- S_i is the source term associated with the ith cell, and
- H_{ij} is the outward normal flux of material across the ijth edge.

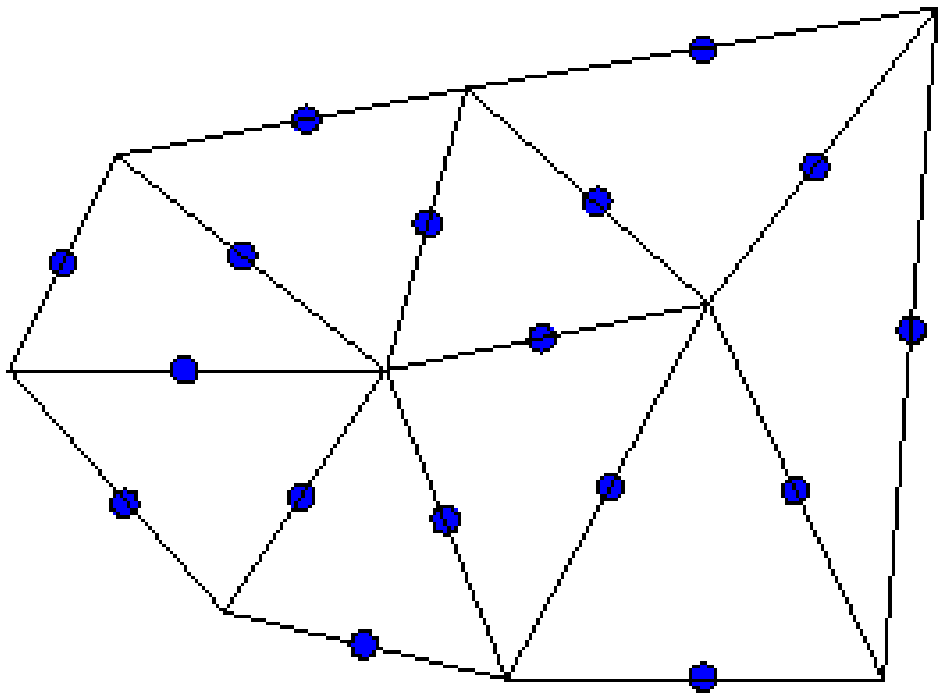
Update Step

```
def evolve_one_euler_step(self, yieldstep, finaltime):
    """
    One Euler Time Step
    Q^{n+1} = E(h) Q^n
    """

    # Compute fluxes across each element edge
    self.compute_fluxes()

    # Update timestep to fit yieldstep and finaltime
    self.update_timestep(yieldstep, finaltime)

    # Update conserved quantities
    self.update_conserved_quantities()

    # Update ghosts
    self.update_ghosts()

    # Update time
    self.time += self.timestep

    # Update vertex and edge values
    self.distribute_to_vertices_and_edges()

    # Update boundary values
    self.update_boundary()
```

Step 1 of 5: Compute fluxes

```
self.compute_fluxes()
```

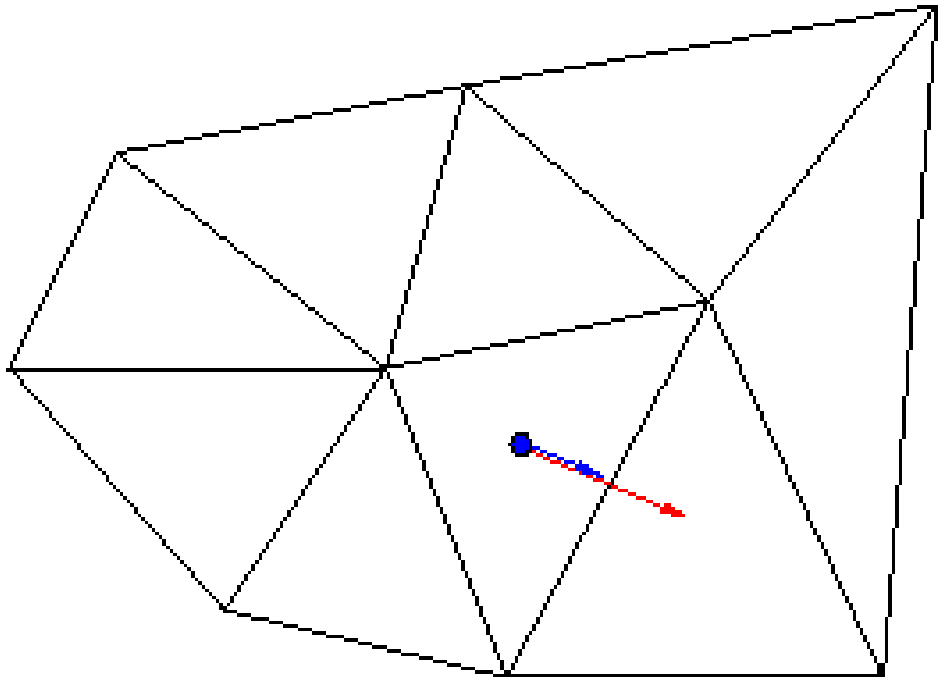
- Compute fluxes across edges using the Central Scheme
- Problem specific flux scheme applied here



Step 2 of 5: Time Step

```
self.update_timestep()
```

- Work out speeds from flux calculation
- Calculate global time step (communication step) to ensure information never skips a triangle

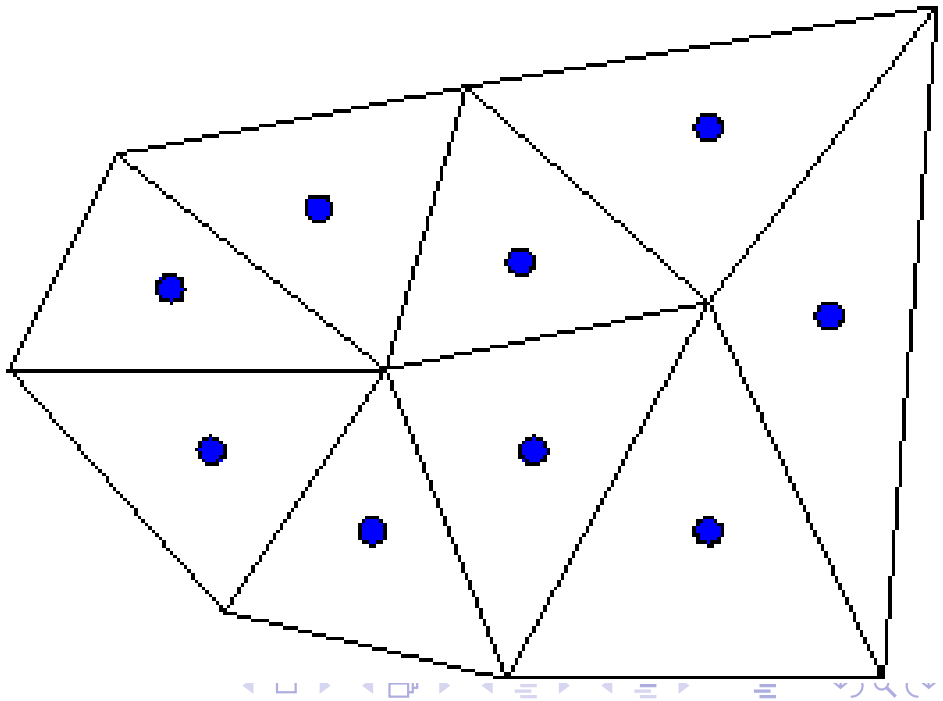


Step 3 of 5: Update Quantities

```
self.update_conserved_quantities()
```

$$U_i^{n+1} = U_i^n - \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} H_{ij} l_{ij} + S_i$$

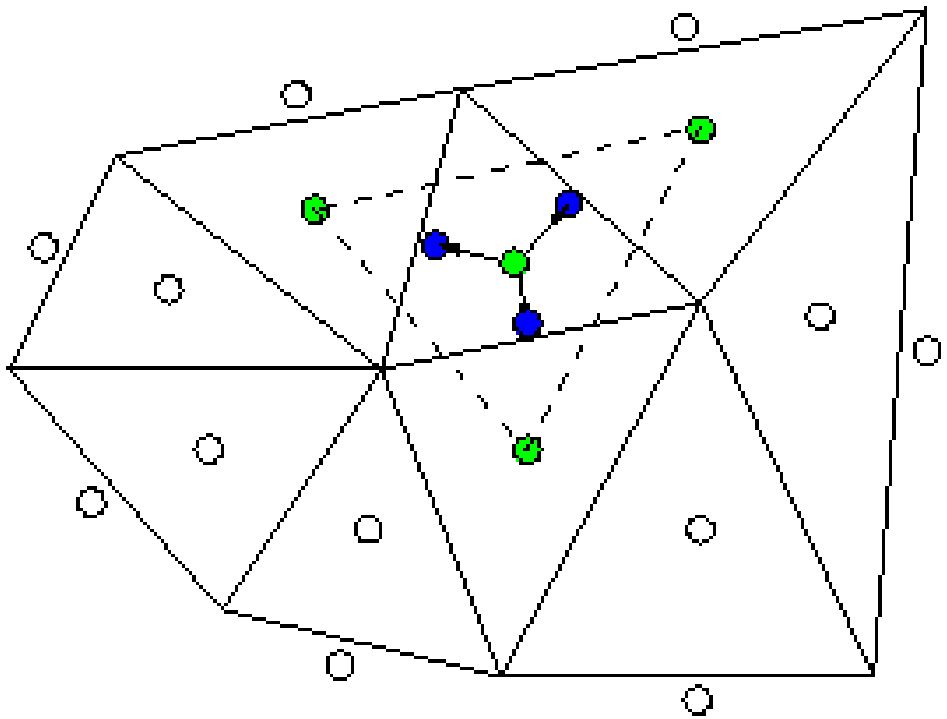
- Update Quantities at Centroids
  - Apply Gravity
  - Apply Bed friction
  - Apply Wind stress
  - Apply Pressure gradients
  - Apply Other physical terms



Step 4 of 5: Compute Quantities at Edges

```
self.distribute_to_vertices_and_edges()
```

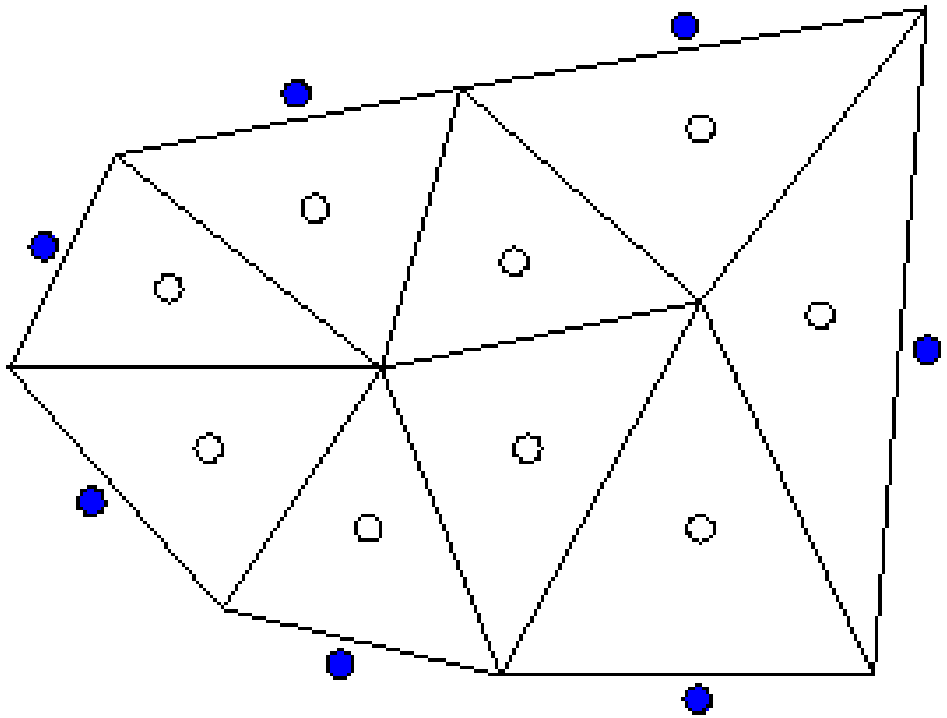
- Use values in neighboring triangles and center to extrapolate
- Limit slope to avoid overshoot
- Ensure water level does not violate bathymetry
- Need to get information from neighbouring triangles



Step 5 of 5: Update boundary values

`self.update_boundary()`

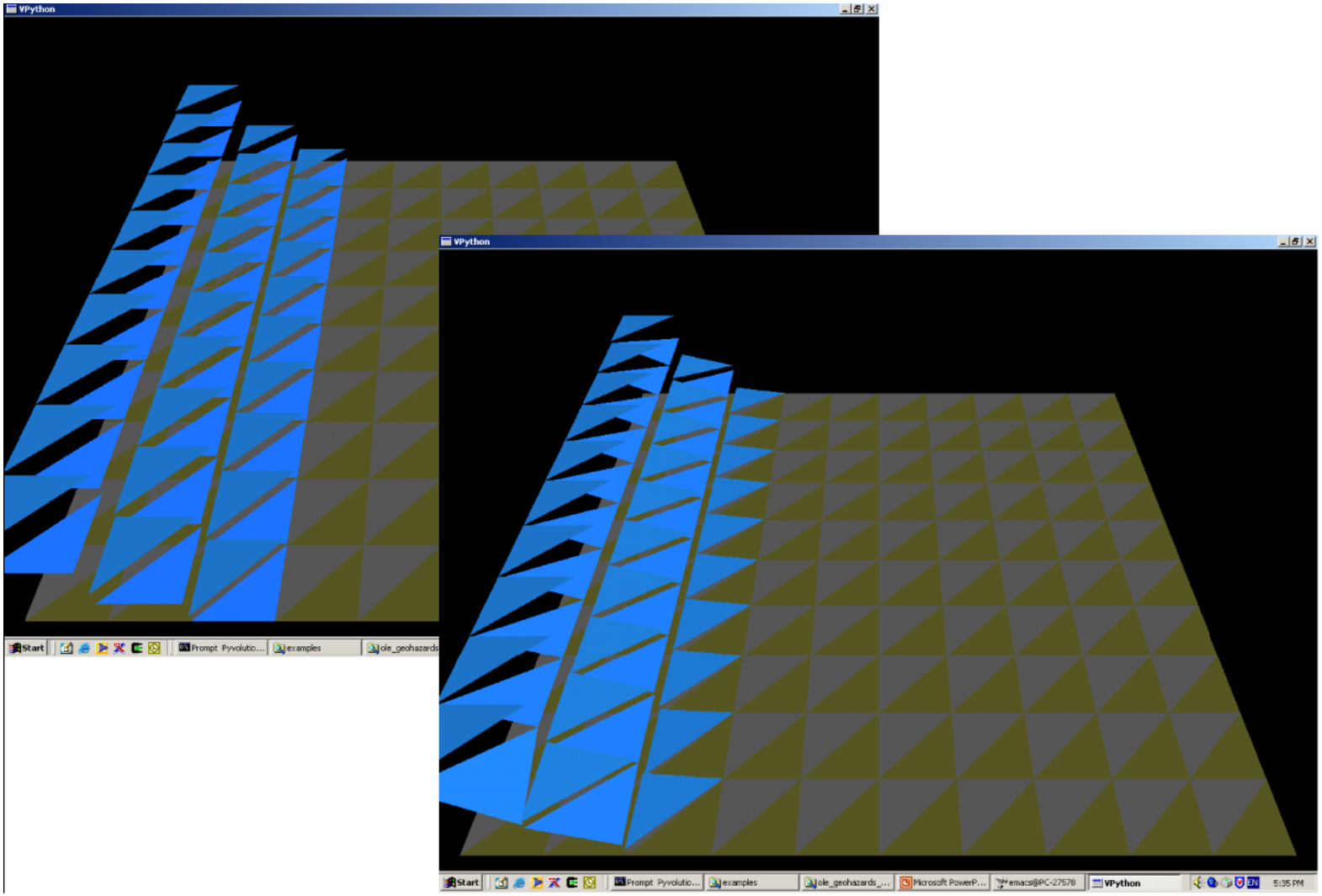
- Using edge values calculate boundary values
  - Dirichlet
  - Reflective
  - Time dependent
  - Spatio-Temporal e.g. output from other simulations
  - Other e.g. periodic



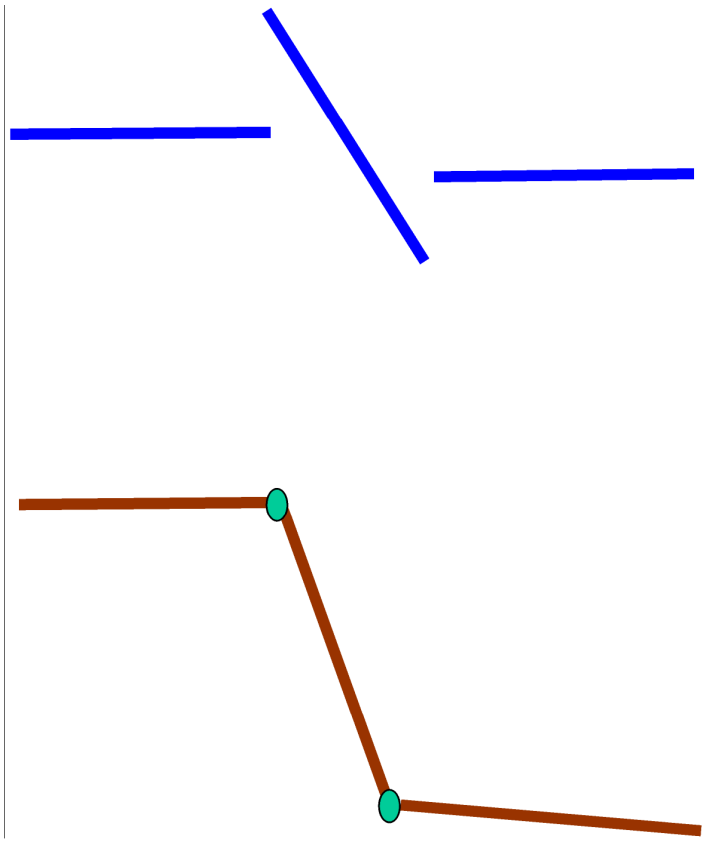
Temporal Updates

```
def evolve_one_euler_step(self, yieldstep, finaltime):
    """
    One Euler Time Step
    Q^{n+1} = E(h) Q^n
    """

def evolve_one_rk2_step(self, yieldstep, finaltime):
    """
    One 2nd order RK timestep
    Q^{n+1} = 0.5 Q^n + 0.5 E(h)^2 Q^n
    """

def evolve_one_rk3_step(self, yieldstep, finaltime):
    """
    One 3rd order RK timestep
    Q^(1) = 3/4 Q^n + 1/4 E(h)^2 Q^n  (at time t^n + h/2)
    Q^{n+1} = 1/3 Q^n + 2/3 E(h) Q^(1) (at time t^{n+1})
    """
```

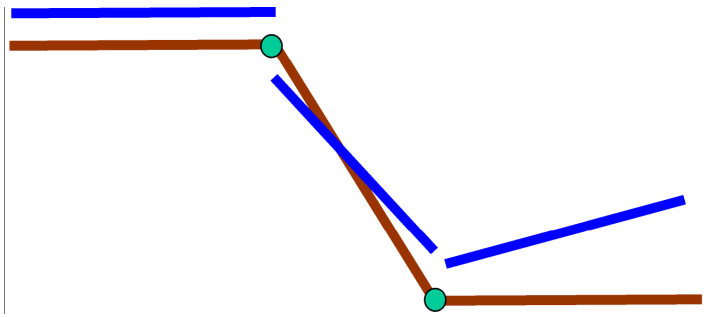
First and Second Order Reconstruction



Limiting on Depth or Stage?

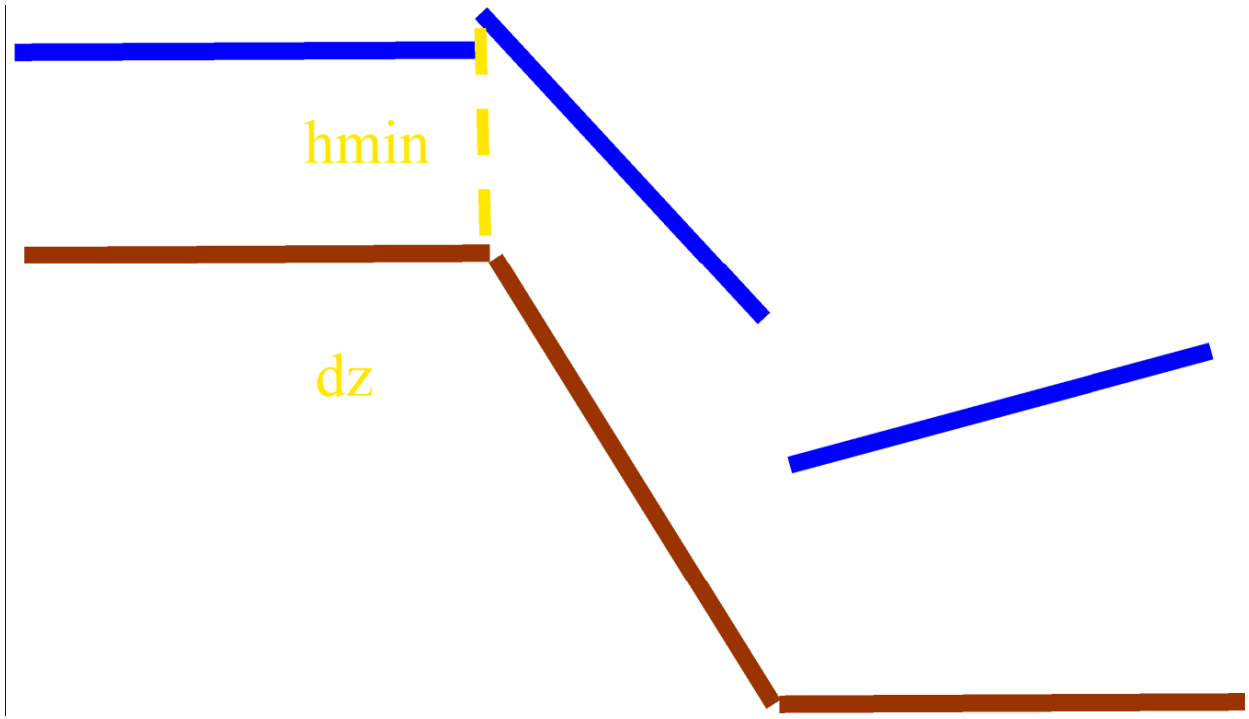


Pure depth limiter may cause wobbles in deep water



Pure stage limiter may cause negative depths

Balanced Limiter



AnuGA Capabilities

- CAN seamlessly model the process of wetting and drying as water advances upon and recedes from the area of inundation.
- Suitable for simulating:
  - water flow onto a beach or dry land
  - flow around structures such as buildings
  - hydraulic shocks (sudden jumps in water level)
- CAN'T model vertical velocities or breaking waves.

AnuGA Software Development

- Written in Python
- Object Oriented
- Bottlenecks written in C
- Overall framework is general
- Open source components
- Automated unit testing
- Subversion for revision control
- Issue tracking (TRAC)
- Elements of agile project management