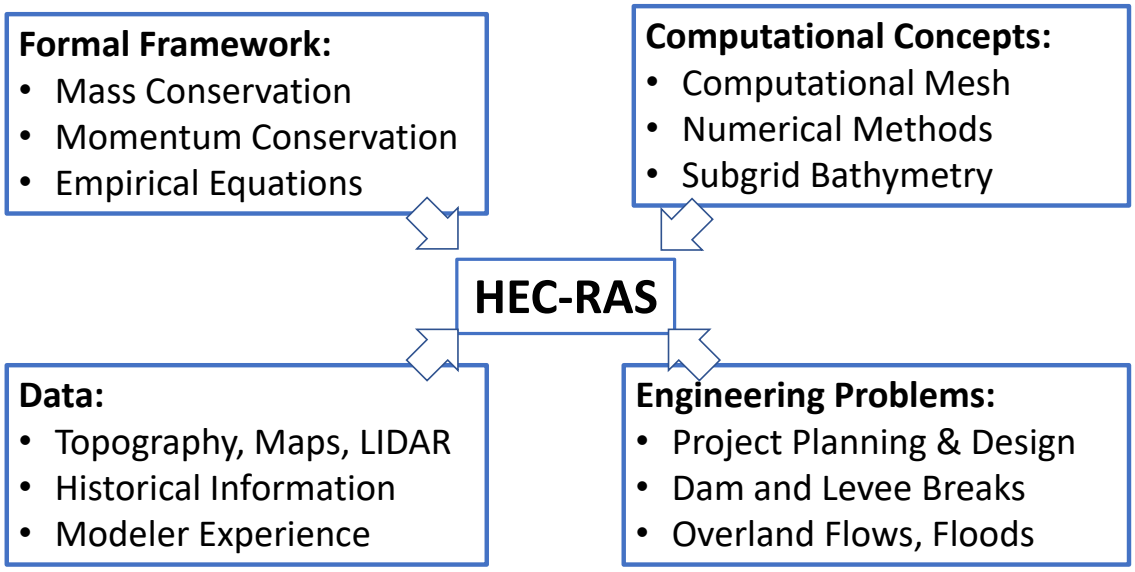
Introduction to 2D Hydraulics Equations

Alex Sánchez, Ph.D.

Senior Hydraulic Engineer
USACE, Institute for Water Resources, Hydrologic Engineering Center

Hydraulic Modeling



2

Hydraulic Modeling is a multidisciplinary field concerned with fluid flow and hydraulic behavior.

As a Hydraulic Modeling software. HEC-RAS is involved with aspects of other areas such as Fluid Mechanics, Data Processing, and Engineering.

This presentation will cover the conceptual framework of HEC-RAS, emphasizing hydrodynamic concepts like mass and momentum balance, friction, and turbulence. Computational concepts including the mesh, numerical methods, and subgrid bathymetry will are covered in detail.

Outline

- Mass Conservation (Continuity)
- Momentum Conservation (Depth-Averaged)
  - Acceleration
  - Coriolis term
  - Hydrostatic pressure
  - Turbulent mixing
  - Friction
- Diffusion Wave Equation
- Numerical Methods

3

Mass Conservation

- Assuming a constant water density

$$\underbrace{\frac{\partial h}{\partial t}}_{\text{Storage}} + \underbrace{\nabla \cdot (h\boldsymbol{V})}_{\text{Flow Divergence}} = \underbrace{q}_{\text{Sourced and sinks}}$$

- Integrating over a computational cell

$$\frac{\partial}{\partial t}\iiint_{\Omega} d\Omega + \iint_{S} (\boldsymbol{V} \cdot \boldsymbol{n}) dS = Q$$

- Finite-Volume Discretization

$$\frac{\Omega_i^{n+1} - \Omega_i^n}{\Delta t} + \sum_{k \in \mathrm{K}(i)} (\boldsymbol{V}_k \cdot \boldsymbol{n}_{ik}) A_k = Q_i$$

h : Water depth

q : Water souce/sink

Ω_i : Cell water volume

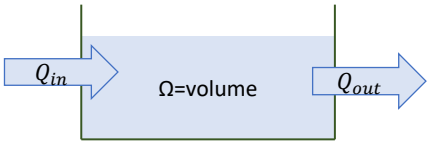
A_k : Face area

$\boldsymbol{V}_k$: Face velocity

$\boldsymbol{n}_{ik}$: Outward face-normal unit vector

Δt : Time step

Change in volume equals sum of inflow and outflow



4

- The most important principle we want enforce is mass conservation.
- By assuming constant water density, the mass conservation can be written in terms of water volume which is referred to as the Continuity Equation.
- It basically means that within a control volume the change in water volume is balanced by the flows coming in and out.
- The bottom equation shows a simple Finite-Volume discretization of the Continuity Equation

Momentum Conservation

- Momentum Equation (non-conservative form)

$$\underbrace{\frac{\partial \boldsymbol{V}}{\partial t}}_{\text{Temporal}} + \underbrace{(\boldsymbol{V}\cdot\nabla)\boldsymbol{V}}_{\text{Advection}} + \underbrace{f_c \boldsymbol{k}\times\boldsymbol{V}}_{\text{Coriolis}} = \underbrace{-g\nabla z_s}_{\text{Pressure gradient}} + \underbrace{\frac{1}{h}\nabla\cdot(\nu_t h\nabla\boldsymbol{V})}_{\text{Diffusion}} - \underbrace{\frac{\boldsymbol{\tau}_b}{\rho R}}_{\text{Bottom Friction}} + \underbrace{\frac{\boldsymbol{\tau}_s}{\rho h}}_{\text{Wind Stress}}$$

- From Newton's 2nd Law of motion
- Assumes constant water density, small vertical velocities, hydrostatic pressure, etc.
- Non-linear and a function of both velocity and water levels
- Continuity and Momentum Equations are the Shallow Water Equations or sometimes referred to as the "Full Momentum" equations in HEC-RAS

$\boldsymbol{V}$: Velocity
z_s : Water level
g : Gravity
ν_t : Turbulent eddy viscosity
h : Water depth
R : Hydraulic Radius
f_c : Coriolis Parameter
τ_b : Bed shear stress
τ_s : Surface stress

5

- The momentum equation is basically Newton's second Law written here with the acceleration on the left and the forces on the right.
- Obviously, the equation is depth-averaged, but it also has some assumptions like constant water density, and hydrostatic pressure, and others.
- The momentum equation is non-linear and a function of both velocity and water levels. So, it's coupled to the continuity equation.
- The system of Continuity and Momentum Equations is the Shallow Water Equations. In HEC-RAS they are sometimes referred to as the "Full Momentum" equations.

Acceleration and Total Derivative

- **Eulerian:** Frame of reference fixed in space and time

$$\frac{\partial \boldsymbol{V}}{\partial t} + (\boldsymbol{V} \cdot \nabla)\boldsymbol{V}$$

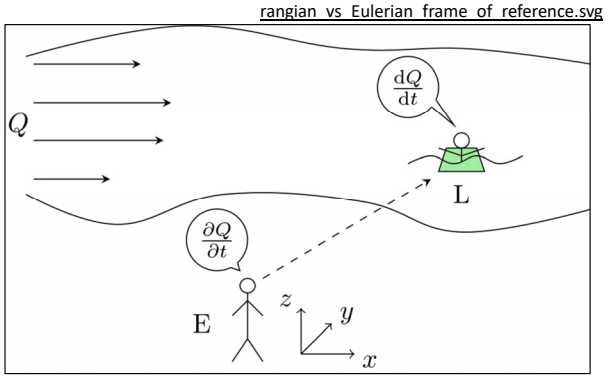
  - Easier to compute
  - Time-step restricted by Courant condition

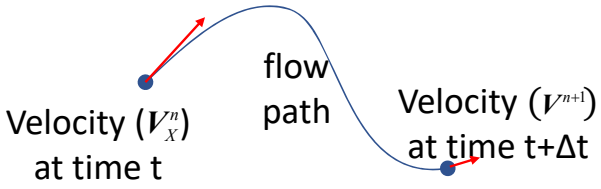
- **Lagrangian:** Frame of reference moves with total derivative along flow path

$$\frac{\partial \boldsymbol{V}}{\partial t} + (\boldsymbol{V} \cdot \nabla)\boldsymbol{V} = \frac{D\boldsymbol{V}}{Dt} = \frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}_X^n}{\Delta t}$$

  - More expensive to compute
  - Allows larger time-steps

https://commons.wikimedia.org/wiki/File:Lagrangian_vs_Eulerian_frame_of_reference.svg





6

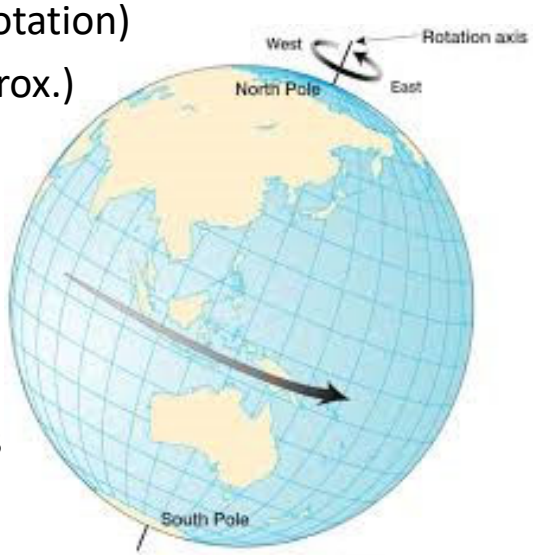
- HEC-RAS has 2 solvers for the shallow water equations: Eulerian (SWE-EM) and the Eulerian-Lagrangian (SWE-ELM) approach.
- They differ in how they treat the acceleration terms.
- In the Eulerian approach the frame of references is fixed in space and time.
- It's easier to compute but it's time step is limited by the Courant condition.
- In the Lagrangian approach to the acceleration terms, the frame of reference moves along a flow path.
- It's more expensive to compute because but allows for larger time steps.

Coriolis Acceleration

- Effect of rotating frame of reference (earth's rotation)
- Constant for the each 2D domain (f-plane approx.)

$$f_c = 2\,\omega \sin\varphi$$

  - ω: sidereal angular velocity of the Earth
  - φ: latitude. Positive for northern hemisphere. Negative for southern hemisphere
- Coriolis acceleration disabled by default to save computational time
- Negligible for most river and flood simulations
- When to enable Coriolis term?
  - Large domains
  - Higher latitudes



7

- The Coriolis acceleration is a result of the earths' rotation.
- It allows for the conservation angular momentum within the rotating reference frame of the earth.
- It's the reason why storms spin counter-clockwise in the Northern hemisphere and clock-wise in the southern hemisphere.
- The Coriolis parameter is a function of latitude, but most RAS models are small enough that an average latitude can be used to compute a constant Coriolis parameter for each 2D area.
- This is known as an f-plane approximation.
- Coriolis is disabled by default, and for most in rivers and flood simulations the Coriolis effect is negligible.
- However, it can be important for large domains or higher latitudes.

Hydrostatic Pressure

- Assumes vertical water accelerations are small compared to gravity
- Total pressure is

$$P = P_{atm} + \rho g\left(z_s - z\right)$$

  - P_{atm}: atmospheric pressure (assumed to be constant)
  - ρ : constant water density
  - g: gravity acceleration constant
  - z_s: water surface elevation
  - z: vertical coordinate
- Pressure gradient

$$\frac{\partial P}{\partial x} = \rho g \frac{\partial z_s}{\partial x}$$

8

- Hydrostatic pressure assumes that the vertical water accelerations are small compared to gravity and this is true for most RAS applications.
- The total pressure is the sum of the atmospheric pressure and the weight of the fluid above a point.
- Since atmospheric pressure is assumed constant, so the horizontal pressure gradient only includes the gradient of the water surface.

Diffusion of Momentum

- Non-conservative Formulation
  - Only option in Version 5.0.7 and earlier,
  - Optional in Version 6.0

$$\frac{D\boldsymbol{V}}{Dt} = -g\nabla z_s + \boldsymbol{\nu}_t \Delta \boldsymbol{V} - \frac{\boldsymbol{\tau}_b}{\rho R}$$

- Conservative Formulation
  - Default in Version 6.0
  - Only option for Eulerian SWE solver

$$\frac{D\boldsymbol{V}}{Dt} = -g\nabla z_s + \frac{1}{h}\nabla \cdot \left(\boldsymbol{\nu}_t h \nabla \boldsymbol{V}\right) - \frac{\boldsymbol{\tau}_b}{\rho R}$$

$\Delta = \nabla^2$: Laplacian

u_N : Face-normal velocity

ν_t : Turbulent eddy viscosity

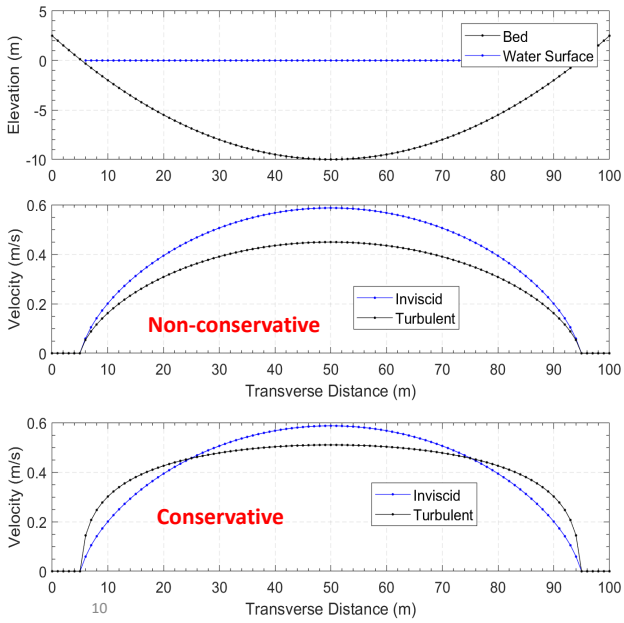
h : Water depth

c_f : Non-linear friction coefficient

9

- Diffusion of Momentum is caused by turbulent mixing and dispersion.
- RAS has 2 formulations for the diffusion term, also known as the mixing term.
- The non-conservative formulation was the only option in versions 5.0.7 and earlier and is still an option in 6.0.
- It computes the Laplacian of the velocity field times eddy viscosity. For many natural flows, the Laplacian tends to be on average negative and results in a net dissipation which is not good because it can start to increase water levels and then in order to calibrate the bottom roughness can be adjusted to compensate.
- This why new Conservative formulation has been developed.
- It's the default in version 6.0 and is the only option for the Eulerian SWE solver.
- It computes the divergence of the diffusive fluxes and for this reason it is more conservative.

Mixing Term Formulation Comparison



Bathymetry and water level

Produces a net dissipation

Decreases velocities in middle of channel but increases velocities near banks

10

- This slide shows a comparison of the two mixing term formulations for a simple parabolic channel.
- The top plot shows bathymetry and water level.
- The middle plot shows the velocity with the non-conservative formulation in black. The blue line represents the velocity profile without mixing. Because the velocity curvature is negative everywhere, the mixing term results in a net dissipation and lower velocities.
- The bottom plot shows the velocity with the conservative formulation also in black and with the blue line representing the velocity without mixing for comparison. There the velocities are decreased along the center of the channel and increased along the sides which is much more realistic.

Eddy Viscosity Model

- Old: Parabolic $\nu_t = Du_*h$
  - Versions 5.0.7 and earlier
  - Isotropic (same in all directions)
  - 1 parameter: mixing coefficient D
- New: Parabolic-Smagorisnky

$$\boldsymbol{v}_t = \boldsymbol{D}u_*h + \left(C_s\Delta\right)^2\left|\overline{S}\right|$$

$$\left|\overline{S}\right| = \sqrt{2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2} \qquad \boldsymbol{D} = \begin{bmatrix} D_{xx} & 0 \\ 0 & D_{yy} \end{bmatrix}$$

$$D_{xx} = D_L\cos^2\theta + D_{,T}\sin^2\theta$$
$$D_{yy} = D_L\sin^2\theta + D_T\cos^2\theta$$

  - Default method in Version 6.0
  - Non-Isotropic (not the same in all directions)
  - 3 parameters: D_L, D_T, and C_s

11

u_* : Shear velocity
h : Water depth
D : Mixing coefficient
D_L : Longitudinal mixing coefficient
D_T : Transverse mixing coefficient
C_s : Smagorinsky coefficient

- One significant improvement in HEC-RAS 6.0 is the addition of a new eddy viscosity model.
- In versions 5.0.7 and earlier, the eddy viscosity was computed with a simple isotropic parabolic model. It has one parameter the mixing coefficient D.
- In version 6.0 there is a new formulation which is a combined Parabolic-Smagorinsky model. It has a non-isotropic mixing coefficient and an additional Smagorinsky term which is a function of the horizontal velocity gradients. This is important because areas with high horizontal shears produce more turbulence.
- The new eddy viscosity model better but it is more computationally expensive because of the velocity gradients.
- It also has 3 parameters instead of 1 so it can be more difficult. However, it does generally produce better results. The 3 parameters are the Longitudinal and Transverse Mixing Coefficients, and the Smagorinsky coefficient.
- If the Longitudinal and Transverse mixing coefficient as set to the same value and the Smagorinsky coefficient is set to zero, then it reduces to the previous turbulence model.
- The Smagorinsky-Lelly model is sometimes criticized because it tends to under-predict the eddy viscosity for fine-resolution meshes.
- However, this model doesn't have that problem because of the first term.
- The first term represents the turbulence produced by bottom friction and dispersion, whereas the second term represents the turbulence produced by subgrid flows and

horizontal shear.

- Both of these formulations are referred as zero-equation turbulence models and they offer a good comprise between accuracy and computational costs.

Bottom Friction

- Resisting force due to relative motion of fluid against the bed
- Bed Shear Stress

$$\boldsymbol{\tau}_b = \rho C_D |\boldsymbol{V}| \boldsymbol{V}$$

- Drag Coefficient

$$C_D = \frac{gn^2}{R^{1/3}}$$

- Friction coefficient

$$c_f = \frac{C_D}{R}|\boldsymbol{V}| = \frac{gn^2}{R^{4/3}}|\boldsymbol{V}|$$

n :Manning coefficient

ρ: water density

g: gravity acceleration constant

$|\boldsymbol{V}|$: velocity magnitude

R: hydraulic radius

12

- Bottom friction is the resisting force due to the relative motion of the fluid against the bed.
- The nonlinear bottom friction coefficient is computed as a function of the Manning's roughness coefficient, gravity, the velocity magnitude, and the hydraulic radius.

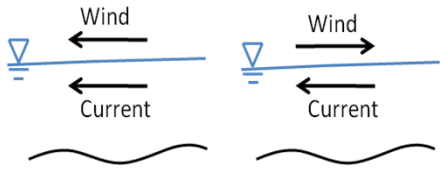
Wind Stress

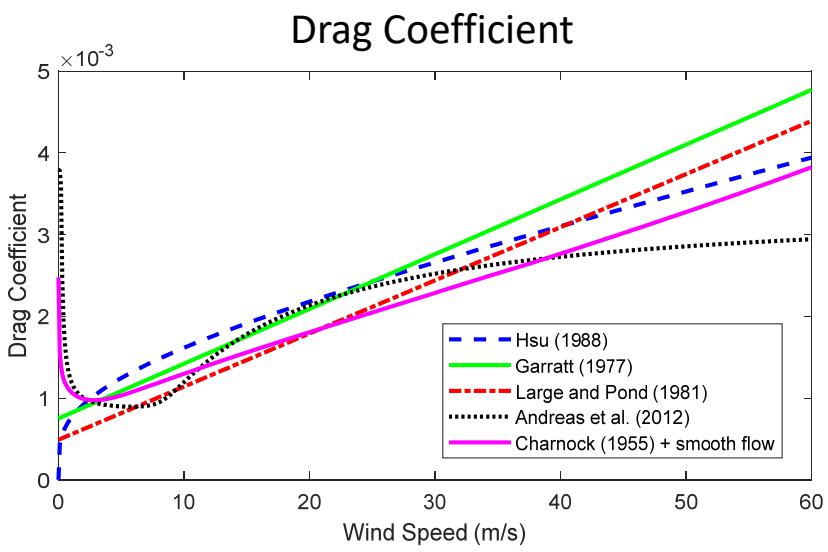
- Surface Stress is given by

$$\boldsymbol{\tau}_s = \rho_a C_D \left| \boldsymbol{W}_{10} \right| \boldsymbol{W}_{10}$$

- Wind Reference Frame

$$\boldsymbol{W}_{10} = \begin{cases} \boldsymbol{W}_{10}^{E} - \boldsymbol{V} & \text{for Lagrangian} \\ \boldsymbol{W}_{10}^{E} & \text{for Eulerian} \end{cases}$$





13

- Wind forcing is new feature 6.0.
- The wind surface stress is computed as a function of the atmospheric density, a drag coefficient, and the 10-m wind velocity.
- Several options are available to compute the wind drag coefficient.
- There is also an option to choose between Eulerian and Lagrangian reference frames.
- The Lagrangian reference frame takes into account the relative motion between the air and water whereas the Eulerian reference frame ignores the water velocity.
- In general, it always recommended to use the Lagrangian reference frame because it's more physically accurate and stable.

Diffusive-Wave Approximation

- Ignoring the following terms

$$\underbrace{\frac{\partial \boldsymbol{V}}{\partial t}}_{\text{Temporal}} + \underbrace{(\boldsymbol{V}\cdot\nabla)\boldsymbol{V}}_{\text{Advection}} + \underbrace{f_c \boldsymbol{k}\times\boldsymbol{V}}_{\text{Coriolis}} = \underbrace{-g\nabla z_s}_{\text{Pressure gradient}} + \underbrace{\frac{1}{h}\nabla\cdot\left(\nu_t h\nabla\boldsymbol{V}\right)}_{\text{Diffusion}} - \underbrace{\frac{\boldsymbol{\tau}_b}{\rho R}}_{\text{Bottom Friction}} + \underbrace{\frac{\boldsymbol{\tau}_s}{\rho h}}_{\text{Wind Stress}}$$

- Expanding and dividing both sides by the square of its norm leads to

$$\boldsymbol{V} = -\alpha\nabla z_s \qquad \alpha = \frac{R^{2/3}}{n\left|\nabla z_s\right|^{1/2}}$$

- Inserting the above equation into the Continuity Equation leads to the Diffusion-Wave Equation (DWE)

$$\frac{\partial h}{\partial t} = \nabla\cdot\left(\alpha h\nabla z_s\right) + q$$

14

- What's very convenient about this equation is that it is no longer a function of current velocities. It only has water levels. Therefore, velocities can be computed once a solution for the water surface is obtained.
- The DWE is simpler and more computationally efficient to solve compared to the SWE's.
- However, the DWE has a more limited applicability than the SWE's.

SWE vs. DWE

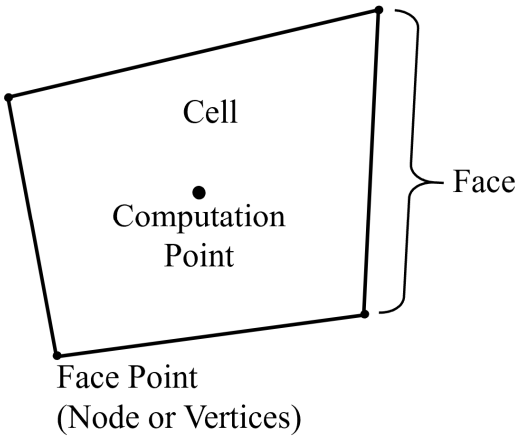
- Use SWE for:
  - Flows with dynamic changes in acceleration
  - Studies with important wave effects, tidal flows
  - Detail solution of flows around obstacles, bridges or bends
  - Simulations influenced by Coriolis, mixing, or wind
  - To obtain high-resolution and detailed flows
- Use DWE for:
  - Flow is mainly driven by gravity and friction
  - Fluid acceleration is monotonic and smooth, no waves
  - To compute approximate global estimates such as flood extent
  - To assess approximate effects of dam breaks
  - To assess interior areas due to levee breeches
  - For quick estimations or preliminary runs

15

- These are some general guidelines for when to use the Diffusion-Wave or Shallow Water Equations.
- But because it is so easy to switch from Diffusion Wave to Shallow Water and RAS is relatively fast. The best practice is to try them both compare.
- I always recommend starting a project with diffusion-wave because it's faster and more stable. Then once the model is running well, try the Shallow Water Equations and compare. This is the most objective and evidence-based way of knowing.

Computational Mesh

- Mesh/grid can be unstructured
- Polygonal cells of up to 8 sides
- Cells must be concave
- Multiple 2D mesh can be run together or independently
- Grid Notation
  - Cells, Faces, Face Points (i.e. nodes or vertices), Computational Points, etc.
- State Variables
  - Cell Water levels
  - Face-normal Velocities



16

- HEC-RAS uses an unstructured polygonal mesh.
- The polygons or cells consist of sides or faces, connected by Face Points (or better-known Nodes or Vertices).
- Cells can have up to 8 faces.
- Cells must be concave.
- Multiple 2D meshes can be run together or independently.
- Each cell has a Computation Point which may or may not be the Cell Centroid.
- In version 6.0 the Computation Points is only used for mesh generation and editing and not in the model computations. Instead, the Cell Centroid is used in the computations.

Numerical Methods

- Both DWE and SWE solvers are **Semi-implicit**
- Terms treated as:
  - Explicit: acceleration and diffusion terms
  - Semi-implicit: friction, flow divergence terms, and water level gradient
  - Fully-Implicit: pressure gradient term (for $\theta = 1$)
- By treating the “fast” pressure gradient term implicitly, the time step limitation based on the wave celerity can be removed
- Both DWE and SWE use **Finite-Difference** and **Finite-Volume** Methods
- Time integration: **Finite-Difference**
- Continuity Equation: **Finite-Volume**
- Momentum Equation: **Finite-Difference** (no control volume)

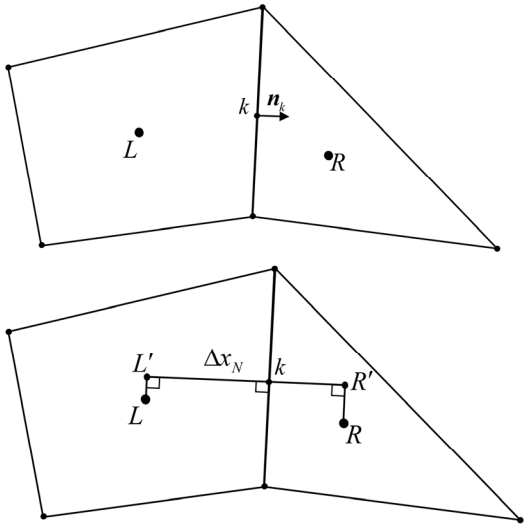
17

Face Water Surface Gradient

- Face-Normal Gradient

$$\nabla z_s \cdot \boldsymbol{n}_k = \frac{\partial z_s}{\partial N} \approx \frac{z_{s,R} - z_{s,L}}{\Delta x_N}$$

  - Uses **Cell Centroids** and **NOT** the **Computation Points**
  - Future versions may include non-orthogonal
  - Compact two-point stencil is computationally efficient and robust
  - Important to have a good quality mesh to reduce errors



18

- The face-water surface gradient calculation has changed in version 6.0.
- The change is small and in fact for Cartesian cells, the schemes are identical. But for other cells they can vary and for highly distorted meshes, they can vary significantly.
- The gradient is computed with a simple two-point stencil.
- The approach uses the orthogonal distance between Cell Centroids and NOT the distance between Computation Points and there is no correction for non-orthogonality.
- We may include corrections in future versions as an option.
- However, the new 2-point stencil is efficient and robust.
- We're also going to working mesh quality to improve the model accuracy.

Momentum Conservation

- Momentum conservation is directionally invariant
- Only "face-normal" component is needed at faces so

$$\frac{\partial u_N}{\partial t} + (\boldsymbol{V} \cdot \nabla) u_N - f_c u_T = -g \frac{\partial z_s}{\partial N} + \frac{1}{h} \nabla \cdot (\nu_t h \nabla u_N) - \frac{\tau_{b,N}}{\rho R} + \frac{\tau_{s,N}}{\rho h}$$

where u_N is the velocity in the N direction

19

- Because momentum conservation is directionally invariant, we can simply things and only solve the momentum equation in the directions that need and that is the face-normal directions.
- The tangential face velocities could be solved with a momentum equation just like the face-normal velocities.
- However, it's much more efficient to reconstruct them from the face-normal velocities.
- This is much more computationally efficient and is very robust.

Face-Tangential Velocity

- Tangential velocities are computed on left and right of face with a Least-squares Formulation

$$S_R = \sum_{k \in R}^{3} \left(\boldsymbol{V}_R \cdot \boldsymbol{n}_k - (u_N)_k \right)^2 \qquad S_L = \sum_{k \in L}^{3} \left(\boldsymbol{V}_L \cdot \boldsymbol{n}_k - (u_N)_k \right)^2$$

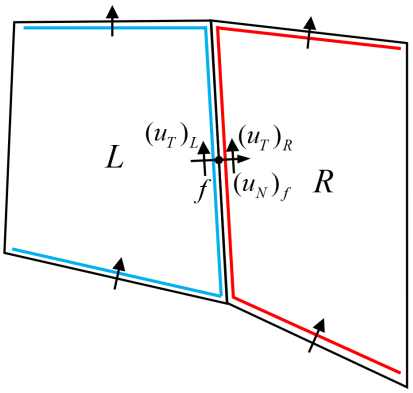
- Of the left and right reconstructed velocities, only the tangential component is used, because the normal component is known

$$(u_T)_R = \boldsymbol{V}_R \cdot \boldsymbol{t}_f \qquad (u_T)_L = \boldsymbol{V}_L \cdot \boldsymbol{t}_f$$

- Average face-tangential velocity computed as

$$(u_T)_f = \frac{(u_T)_R + (u_T)_L}{2}$$

Double-C Stencil



20

- The face-tangential velocities are computed with what we call the Double-C stencil approach.
- The method computes 2 face-tangential velocities, one on each side of a face.
- A full velocity vector is reconstructed on the blue and red C-stencils shown in the figure by solving a least-squares problem. The face-normal component is then thrown out, because it's already known, and the tangential component is applied at the face.
- If the 2 cells are hydraulically connected, the average face-tangential velocity is computed as a simple average of the left and right tangential velocities.
- This same double-C stencil approach is used to compute the face-tangential water surface gradient utilized by the DWE.

Discretization

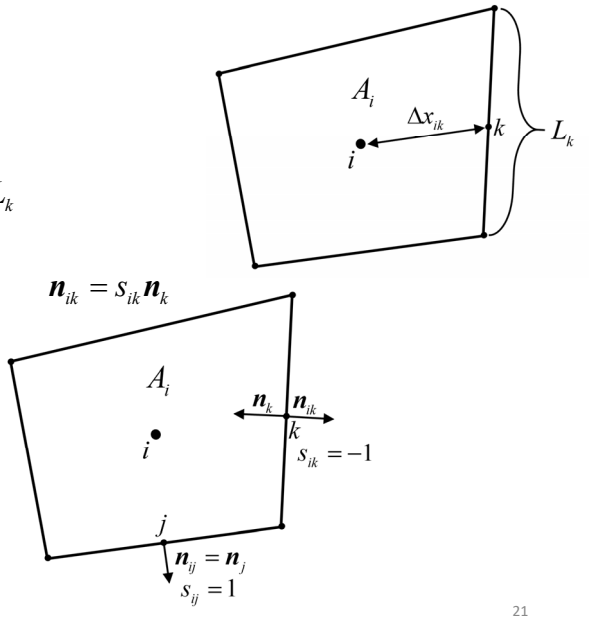
- Cell Velocity Gradient (x-direction)
  - Gauss' Divergence Theorem

$$\nabla u_i = \frac{1}{A_i}\int_A \nabla u \mathrm{d}A = \frac{1}{A_i}\oint_L u\boldsymbol{n}\mathrm{d}L = \frac{1}{A_i}\sum_{k\in i} u_k \boldsymbol{n}_{ik} L_k$$

  - Needed tor turbulence modeling
- Cell Velocity
  - Perot's Method

$$\boldsymbol{V}_i = \frac{1}{A_i}\sum_{k\in i} \Delta x_{ik} L_k \boldsymbol{n}_k (u_N)_k$$

  - Needed for the conservative form of the mixing term and for Eulerian advection



21

- The cell velocity gradients are computed by applying the Divergence Theorem and are needed for turbulence modeling.
- The cell velocities are computed with Perot's method and are needed for the conservative form of the mixing term and for the advection term of the Eulerian Shallow Water Equation solver.

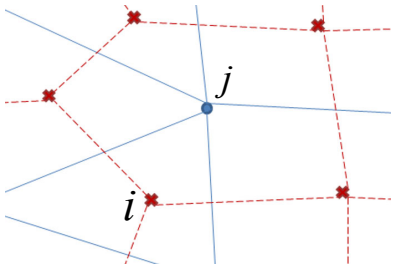
Discretization: Laplacian

- Node Laplacian

$$\left(\nabla^2 \boldsymbol{V}\right)_j = \left[\nabla \cdot \left(\nabla \boldsymbol{V}\right)\right]_j \approx \sum_i d_i \left(\nabla \boldsymbol{V}\right)_i$$

$$\left(\nabla \boldsymbol{V}\right)_i = \sum_k c_k \boldsymbol{V}_k$$

i : Cells
j : Nodes
k : Faces



- Used only by non-conservative turbulence

22

- The Laplacian of the velocity field is computed on the nodes based on the cell velocity gradients.
- The image on the right shows the mesh and dual mesh. The blue circle on the nodes and red X's are the neighboring computation points at which the velocity gradients are computed.
- The Laplacian is computed similar to the velocity gradients by utilizing the divergence theorem but in this case on the dual mesh shown in red.

Backtracking

$$\frac{\partial u_N}{\partial t} + (\boldsymbol{V} \cdot \nabla) u_N = \frac{Du_N}{Dt} \approx \frac{u_N^{n+1} - u_X^n}{\Delta t}$$

$$\frac{d\boldsymbol{x}_P}{dt} = \boldsymbol{V}(\boldsymbol{x}, t)$$

1. Interpolate node velocities from faces
2. Set starting location and remaining time as f and $T_R = \Delta t$
3. From starting location and velocity, find location B
4. Compute time to location B: $T_B = (\boldsymbol{x}_A - \boldsymbol{x}_B)\boldsymbol{V}_A^{-1}$
5. Interpolate velocity at location B: $\boldsymbol{V}_B = w_{n1}\boldsymbol{V}_{n1} + w_{n2}\boldsymbol{V}_{n2}$

if $T_B > T_R$

6. Set $A = B$, $T_R = T_R - T_B$, and go to step 3

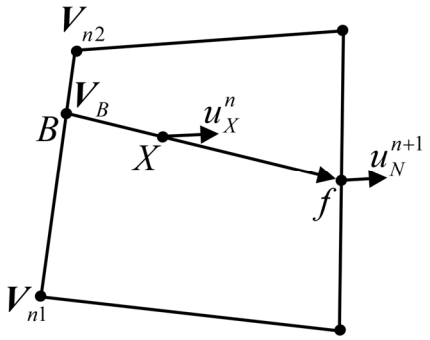
else

7. Find location X as $\boldsymbol{x}_X = \boldsymbol{x}_f - T_R \boldsymbol{V}_A$
8. Interpolate velocity vector at X

$$\boldsymbol{V}_X = T_B^{-1}\left[T_R \boldsymbol{V}_B + (T_B - T_R)\boldsymbol{V}_A\right]$$

9. Compute advective velocity

$$u_X = \boldsymbol{n}_f \cdot \boldsymbol{V}_X$$



23

- The Courant condition is avoided in the Eulerian-Lagrangian solver by incorporating advection in the total derivative.
- An integral part of this is backtracking, which the processes of the tracing back the flow path or characteristic line from a face to its starting location X at the beginning of the time.
- The algorithm is RAS is computationally efficient, robust, and has good time convergence.
- The actual algorithm in RAS is more complicated than this because of boundary conditions and special cases, but this the general idea.

1. First the node velocities are interpolated from the faces using the previous time step velocity field.
2. The backtracking starts at the faces. The starting location A is set to face center f and the remaining backtracking time T_R is initialized as the time step.
3. Next location the location B is computed from the starting location A.
4. Then the time to location B (T_B) is computed
5. Then the velocity at B is interpolated from the nodes

If T_B is greater than the remaining time T_R,

5. The particle has crossed into the neighboring cell. B is set as the new starting location and T_B is subtracted from the remaining time.

If T_B is less than the remaining time T_R, the particle is within the current cell.

6. In this case, the location of X is computed as shown and
7. The velocity at X is interpolated from the velocities at A and B.
8. Finally, the face-normal advective velocity is computed with the dot product of the face unit and V_X.

Coriolis Term

- Eulerian-Lagrangian Solver
  - Fractional Step Method (Semi-implicit)

$$\begin{pmatrix} 1 & \theta\Delta t f_c \\ \theta\Delta t f_c & 1 \end{pmatrix}\begin{pmatrix} u^* \\ v^* \end{pmatrix} = \begin{pmatrix} u_X^n + (1-\theta)\Delta t f_c v_X^n \\ v_X^n + (1-\theta)\Delta t f_c u_X^n \end{pmatrix} \qquad \boldsymbol{V}^* = \begin{pmatrix} u^* \\ v^* \end{pmatrix}$$

$$\frac{u_N^{n+1} - u_N^*}{\Delta t} = -g\frac{\partial z_s^{n+\theta}}{\partial N} - c_f u_N^{n+1} \qquad u_N^* = \boldsymbol{n}_f \cdot \boldsymbol{V}^*$$

- Eulerian Solver
  - Explicit

$$\frac{u_N^{n+1} - u_N^n}{\Delta t} + (V \cdot \nabla)u_N^n - f u_T^n = -g\frac{\partial z_s^{n+\theta}}{\partial N} - c_f u_N^{n+1}$$

24

f : Coriolis Parameter

θ : Implicit weighting factor

Δt : Time step interval

u_X^n : Backtracking velocity

- The Coriolis term is treated slightly differently in the ELM and EM solvers.
- In the ELM solver, Coriolis is computed semi-implicitly with a Fractional Step Method.
- In the first step a 2x2 matrix is inverted analytically to compute an interim velocity vector V star which includes the Coriolis. The second step includes all remaining terms.
- The treatment of Coriolis in the EM solver is much simpler. It doesn't use a fractional step method. Instead, the Coriolis computed explicitly based on the face-tangential velocity. The tangential velocity is computed from a least-squares of the neighboring face-normal velocities.

Eulerian-Lagrangian Momentum Equation

- Semi-discrete form (2nd Fractional Step)

$$\frac{u_N^{n+1} - u_N^*}{\Delta t} = -g\frac{\partial z_s^{n+\theta}}{\partial N} + \left[\frac{1}{h}\nabla\cdot\left(\nu_t h^n \nabla u_N\right)\right]_X^n - c_f u_N^{n+1} + \frac{\tau_{s,N}}{\rho h_f^n}$$

where

$$z_a^{n+\theta} = (1-\theta)z_s^n + \theta z_s^{n+1}$$

$$u_N^* = \boldsymbol{V}^* \cdot \boldsymbol{n}_f$$

- Velocity $\boldsymbol{V}^*$ includes Coriolis
- Mixing term is interpolated at backtracking location X and based on previous time step velocity field
- Friction term is semi-implicit

25

- The second fractional step of the ELM solver is shown here in semi-discrete form.
- The velocity uN* is the face-normal component of the intermediate velocity computed from the first fractional step at the backtracking location X and thefore includes Coriolis.
- The water surface gradient is computed semi-implicitly based on the weighting factor theta.
- The mixing or diffusion term is explicit and Lagrangian. In other words, it's interpolated at the backtracking location X and is based on the previous velocity field.
- Lastly, the friction and pressure gradient terms are treated semi-implicitly.

Eulerian Momentum Equation

- Semi-discrete form

$$\frac{u_N^{n+1}-u_N^n}{\Delta t}+\left(\boldsymbol{V}^n\cdot\nabla\right)u_N^n-fu_T^n=-g\frac{\partial z_s^{n+\theta}}{\partial N}+\left[\frac{1}{h}\nabla\cdot\left(\nu_t h\nabla u_N\right)\right]_f^n-c_f u_N^{n+1}+\frac{\tau_{s,N}}{\rho h_f^n}$$

where

$$z_s^{n+\theta}=(1-\theta)z_s^n+\theta z_s^{n+1} \qquad \bar{h}_f=\alpha_f^L h_L+\alpha_f^R h_R$$

- Coriolis term computed at face f and is explicit
- No fractional step method like ELM solver
- Mixing term is computed at face f and is explicit
- Friction and pressure gradient terms are semi-implicit

26

- The semi-discrete Eulerian Momentum Equation looks similar to the Eulerian-Lagrangian.
- As before, the water surface gradient and friction semi-implicit
- However, here the Coriolis term treated explicitly.
- There's no fractional step method here.
- The mixing is also explicit and Eulerian.

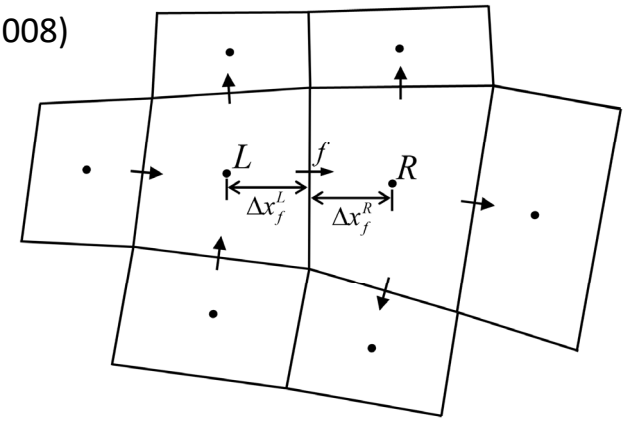
Discretization: Eulerian Advection

- Approach from Kramer and Stelling (2008)

$$(\boldsymbol{V}\cdot\nabla)u_N \approx \frac{\alpha_f^L}{\bar{h}_f A_L}\sum_{k\in L} s_{Lk}Q_k\left[\boldsymbol{V}_k^u\cdot\boldsymbol{n}_f-(u_N)_f\right] + \frac{\alpha_f^R}{\bar{h}_f A_R}\sum_{k\in R} s_{Rk}Q_k\left[\boldsymbol{V}_k^u\cdot\boldsymbol{n}_f-(u_N)_f\right]$$

$$\alpha_f^L=\frac{\Delta x_f^L}{\Delta x_f^L+\Delta x_f^R} \qquad \bar{h}_f=\alpha_f^L h_L+\alpha_f^R h_R$$

$$\alpha_f^R=1-\alpha_f^L$$



- Courant-Freidrichs-Lewy (CFL) Condition

$$C=\frac{U\Delta t}{\Delta x}\le 1$$

27

- Of course, the main difference between the ELM and EM solvers is how they treat advection.
- In the EM solver, advection computed with what I would call is a Finite-Volume Advection Interpolation approach.
- It reminds me of Rhie and Chow's momentum interpolation approach.
- Basically, the advection term is computed on either side of a face and then interpolated onto the face.
- Here the alpha's are the interpolation weights
- A simple first order upwinding scheme is used to compute the face advective velocities.

Discretization: Diffusion Term

- Non-Conservative Form

$$\boldsymbol{\nu}_t \nabla^2 u_N \Big|_f \approx \nu_{t,f}^n \left(\nabla^2 \boldsymbol{V}\right)_X^n \cdot \boldsymbol{n}_f$$

- Conservative Form

$$\frac{1}{h}\nabla\cdot\left(\boldsymbol{\nu}_t h \nabla u_N\right)\Big|_f \approx \frac{\alpha_f^L}{\bar{h}_f A_L}\sum_{k\in L} A_k \nu_{t,k}\frac{\boldsymbol{n}_f\cdot\left(\boldsymbol{V}_j - \boldsymbol{V}_L\right)}{\Delta x_{L,j}} + \frac{\alpha_f^R}{\bar{h}_f A_R}\sum_{k\in R} A_k \nu_{t,k}\frac{\boldsymbol{n}_f\cdot\left(\boldsymbol{V}_j - \boldsymbol{V}_R\right)}{\Delta x_{R,j}}$$

- Discretization same for both ELM and EM solvers
- Approximate Stability Criteria for EM solver $\dfrac{\nu_t \Delta t}{\Delta x^2} \le \dfrac{1}{2}$
- ELM interpolates term to location X

$$\left[\frac{1}{h}\nabla\cdot\left(\boldsymbol{\nu}_t h \nabla u_N\right)\right]_X^n$$

28

- The non-conservative form of the mixing term is only available with the ELM solver.
- It's computed as the explicit face eddy viscosity times the dot product of the velocity Laplacian and the face unit vector.
- The velocity Laplacian is interpolated at the backtracking location X.
- The eddy viscosity could also be backtracked as well, but tests have shown that's its more stable to use the value at the face.
- The Conservative form of the mixing term is available with both the ELM and EM solvers.
- It's computed with a similar weighting scheme as the Eulerian advection
- Here the divergence of the diffusive fluxes are summed around the left and right cell faces to compute a diffusion term on each.
- These are then weighted to compute a value at the face with the same alpha coefficients as the advection term.
- In order to simplify the code, the same discretization is used for both the ELM and EM solvers
- However, for ELM solver the mixing is backtracked and interpolated at the location X.
- This removes the stability criteria from the diffusion term and allows it to use a larger step.

Turbulence Numerical Implementations

- Non-conservative mixing
  - Eulerian-Lagrangian Solver

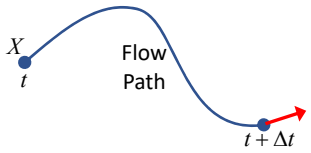
Eulerian-Lagrangian

$$\frac{u_N^{n+1}-u_N^*}{\Delta t}=-g\frac{\partial z_s^{n+\theta}}{\partial N}+\nu_t^n\left(\Delta u_N^n\right)_X-c_f u_N^{n+1}$$

- Conservative mixing
  - Eulerian-Lagrangian Solver

Lagrangian

$$\frac{u_N^{n+1}-u_N^*}{\Delta t}=-g\frac{\partial z_s^{n+\theta}}{\partial N}+\left[\frac{1}{h^n}\nabla\cdot\left(\nu_t^n h^n\nabla u_N^n\right)\right]_X-c_f u_N^{n+1}$$



  - Eulerian Solver

Eulerian

$$\frac{u_N^{n+1}-u_N^n}{\Delta t}+\left(V\cdot\nabla\right)u_N^n-fu_T^n=-g\frac{\partial z_s^{n+\theta}}{\partial N}+\frac{1}{h^n}\nabla\cdot\left(\nu_t^n h^n\nabla u_N^n\right)-c_f u_N^{n+1}$$

29

- All diffusion terms explicitly
- Approximate Stability Criteria for Eulerian Solver

$$\frac{\nu_t\Delta t}{\Delta x^2}\le\frac{1}{2}$$

- In order to understand the strengths and limitations of the different formulations and solvers it's important to understand some basic aspects of the numerical implementations.
- The non-conservative form of mixing remains largely the same in Version 6.0 except for the eddy viscosity. The mixing term is Eulerian-Lagrangian. The eddy viscosity is computed at face while the velocity Laplacian is backtracked. Through testing, this was found better convergence properties then a full Lagrangian approach.
- The non-conservative form of mixing is not available with the EM solver.
- The Conservative Mixing term is available with both the EM and ELM solver.
- In both cases, the mixing term is discretized the same, however in the ELM solver, term is backtracked. This means it's interpolated at the backtracking location X.

Eulerian-Lagrangian vs. Eulerian SWE Solvers

- ELM-SWE
  - Only solver available in V5.0.7 and earlier
  - Default in V6.0
  - Not limited by Courant condition
  - Excellent stability
  - Can have momentum conservation problems around shocks or where the flow changes rapidly
- EM-SWE
  - New to V6.0 as an option
  - Limited to Courant less than 1.0
  - Good Stability
  - Improved momentum conservation for all flow conditions

Strength/Feature/Capability	SWE-ELM	SWE-EM
Larger Time Step	X	
Best Stability	X	
Courant Stability Criteria		X
Diffusion Stability Criteria		X
Computational Speed	X	
Wet/dry > 1 cell per time step	X	
Best Momentum Conservation		X
Non-Conservative Mixing		X
Conservative Mixing	X	X
Wind	X	X

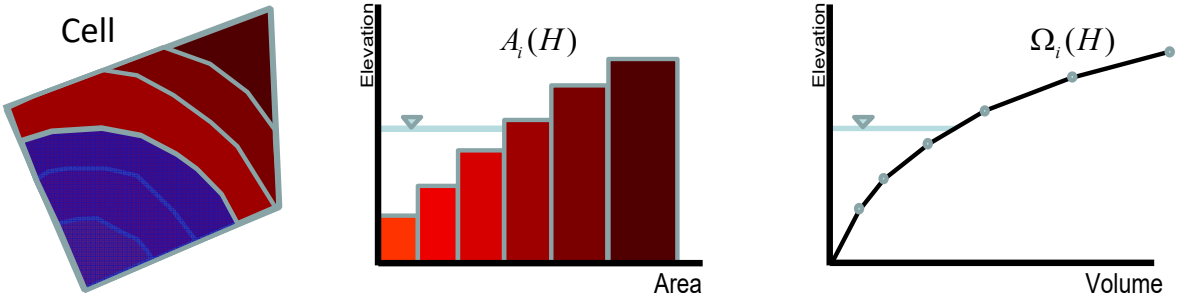
30

Subgrid Modeling

- Problem
  - Water levels usually vary much more smoothly than the terrain
  - Unfeasible to resolve every detail of the terrain with the computational mesh
- Approach
  - Utilize a grid resolution sufficient to resolve the hydraulics
  - Capture the details of the subgrid terrain through hydraulic properties tables

31

Subgrid Bathymetry: Cells

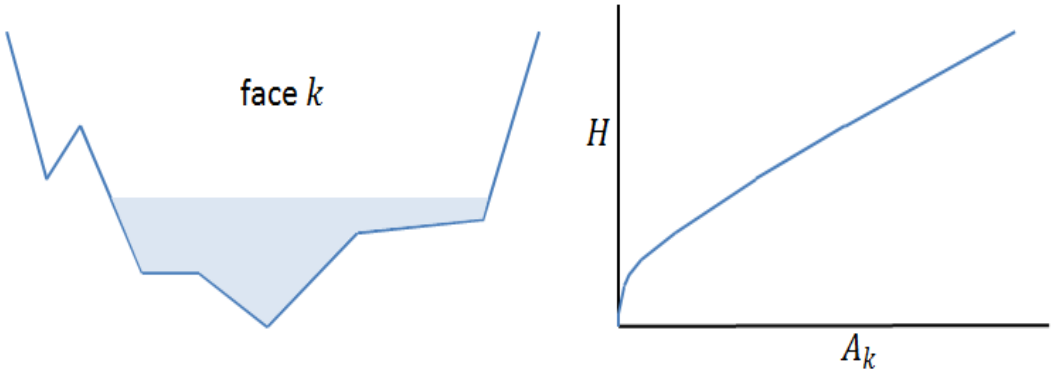


32

- On the left you a cell that is partially wet and dry; the blue region being the area under water.
- The terrain is binned into discrete elevations which can be used to create a curve of horizontal area as a function of elevation (shown in the middle).
- Each bin represents the area corresponding to specific elevations and is used to compute the total wetted cell area for a specific water level.
- The area-elevation curve is integrated to produce and a piece-wise linear volume-elevation curve. This curve is used to compute the cell water volume for any water level.

Subgrid Bathymetry: Faces

- Faces treated similar to cells
- Hydraulic property tables computed
  - Wetted length
  - Wetted Perimeter
  - Area



33

- The subgrid bathymetry at faces is computed similar to cells.
- The property tables computed at faces are the wetted length, wetted perimeter, and wetted vertical area.
- The figure on the left is an example of an irregular bed profile at a face. The light blue shaded region is the wetted area corresponding to a particular water level.
- The figure on the right represents the relationship for the face area and the water level for that face.

Benefits of Subgrid Bathymetry

Shown here is an example of how the computational cells in HEC-RAS contain enough hydraulic detail that flow can move through a channel, even though the channel is smaller than the cell size. In the above example, the cells are 500 ft by 500 ft. Water will move though the channel portion of the cells, because the details of the channel cross sections are contained within the faces. Additionally, the details of the cell elevation-volume curve allow for accurate water storage within the channels. In this type of example, flow can move through a cell/channel in a 1D-type of mode, while flow in the overbank areas will be 2D from cell to cell. If the user wants more detail within the channel, such as 2D flow velocities and varying water surface elevations, then smaller cell size can easily be specified within the channel. However, if the goal is to convey water through the channel and capture the 2D flow the floodplain, then this is a very viable option.

Solution Procedure

- System of equations

$$\boldsymbol{\Omega} + \boldsymbol{\Psi Z} = \boldsymbol{b}$$

- Algorithm
  1. Compute Right-Hand-Side $\boldsymbol{b}$
     - Contains explicit terms: advection, diffusion, wind, etc.
  2. Outer Loop (Assembly and Updates)
     - Update linearized terms and variables including coefficient matrix $\boldsymbol{\Psi}$
  3. Inner Loop (Newton Iterations)

$$\boldsymbol{Z}^{m+1} = \boldsymbol{Z}^{m} - \left[\boldsymbol{\Psi} + \boldsymbol{A}^{m}\right]^{-1}\left(\boldsymbol{\Omega}^{m} + \boldsymbol{\Psi Z}^{m} - \boldsymbol{b}\right)$$

$\boldsymbol{Z}$: Water level

$\boldsymbol{\Omega}$: Water volume

$\boldsymbol{\psi}$: Coefficient matrix

$\boldsymbol{b}$: Right-hand-side

m : Iteration index

$\boldsymbol{A}$: Diagonal matrix of cell wet surface areas

35

- The final system of equations can be written in matrix form as the first equation.
- The computational algorithm begins each time by computing the right-hand-side **b** which contains the sources/sinks and explicit terms.
- The outer loop computes or updates the linearized terms and variables such as the coefficient matrix and wetted surface area.
- The inner loop ensures mass conservation and computes water levels via Newton-like iterations which require solving the sparse matrix shown at the bottom.
- Both the inner and outer loops are repeated until convergence.

Boundary Conditions

- **Stage Hydrograph**. Upstream or downstream
- **Flow Hydrograph**. Upstream or downstream. Local conveyance and velocities computed automatically.
- **Normal Depth BC.** At downstream boundaries.
- **Rating Curve BC.**
- **Wind.** Only for shallow-water equations.
- **Precipitation, evapotranspiration, and infiltration.** Included as sources and sinks in the continuity equation.
- 1D reaches and 2D areas can be connected
- Multiple 2D areas can be connected to each other
- 2D areas can be connected to 1D lateral structures such as levees to simulate levee breaches

36

- The 2D model supports various types of boundary conditions.
- A stage hydrograph can be specified upstream or downstream. Water can flow in and out of stage boundaries and there can even be flow reversals within the same boundary.
- Flow hydrograph can also be specified upstream or downstream but is typically specified upstream. The flow hydrograph BC requires a user-specified energy grade slope which is used to compute a normal depth from a flow rate and the bathymetry profile at the boundary. A conveyance approach is used to distribute the total boundary flow at every face along a boundary.
- A normal depth BC can only be specified at downstream boundaries.
- The rating curve BC specifies the total boundary flow as a function of stage. It can only be specified at downstream (or outflow) boundaries. The total flow is distributed along the wet boundary based on conveyance.
- Wind forcing is new in version 6.0. Wind surfaces stresses are computed from wind velocity data and can only applied with the shallow-water equations.
- Precipitation, evapotranspiration, and infiltration are also new version 6.0. These are applied as sources and sinks within continuity equation.
- 1D reaches and 2D areas can be connected or coupled.
- Multiple 2D areas can be connected to each other.
- And finally 2D areas can be connected to 1D lateral structures such as levees.

Computational Implementation

- Multiple 2D areas can be computed independently and simultaneously
- All solvers are can be run on multiple cores
- 2D solvers and parameters can be selected independently for each 2D area
- A partial grid solution keeps track of active portion of mesh and only computes the solution for active portion significantly reducing computational times.

37

Compute Engine written in Fortran

Theta parameter controls shape of stencil in time: ½ = Crank-Nicolson

Thank You!

HEC-RAS Website:
https://www.hec.usace.army.mil/software/hec-ras/

Online Documentation:
https://www.hec.usace.army.mil/confluence/rasdocs

US Army Corps of Engineers

38