Advanced Parameters in HEC-RAS

Alex Sánchez, PhD

Senior Hydraulic Engineer

USACE, Institute for Water Resources, Hydrologic Engineering Center

1

Overview

- Turbulence
  - Overview of Theory
  - Numerical Implementation
  - Example
  - Guidance and Best Practices
- Coriolis
  - Overview of Theory
  - Numerical Implementation
  - Guidance and Best Practices
- Matrix Solvers
  - Direct vs. Iterative Solvers
  - Iterative Solver Input Parameters
  - Example
  - Guidance and Best Practices

2

- A lot of the material covered in this video is described in detail in the HEC-RAS hydraulic reference and user's manuals.
- These documents are available from the HEC-RAS Help menu or can be downloaded from the HEC-RAS website.

HEC-RAS Turbulence Modeling

- Turbulence is a type of fluid flow which is unsteady, irregular in space and time, dissipative, and diffusive
- Most natural flows are turbulent
- Dispersion is due to the non-uniform velocity distribution
- Turbulence and dispersion are represented with a diffusion term in the momentum equations
- Two aspects (both changed for Version 6.0)
  1. Formulation in momentum equations
  2. Eddy viscosity formulation (aka “turbulence model”)

3

- Turbulence is a type of fluid motion which unsteady, irregular in space and time, dissipative, and diffusive.
- Turbulence modeling is the processes of representing or simulating the turbulent effects within flow models using mathematical and empirical equations.
- Most natural flows are turbulent
- In addition, the current velocities are both depth-averaged and horizontally averaged along faces. The non-uniform velocity distributions in the vertical and horizontal lead to momentum dispersion.
- Both turbulence and dispersion are modeled with a diffusion term in the momentum equations
- There are two aspects in Turbulence Modeling which often get confused:
  1. The formulation of the momentum diffusion term within the Shallow-Water Equations.
  2. How the eddy viscosity calculated. This is referred to as the turbulence model.
- Both of these aspects have changed and are improved in Version 6.0.

Momentum Equations: Diffusion Term

- Non-conservative Formulation
  - Only option in Versions 5.0.7 and earlier
  - Still in option in Version 6.0 and later

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}-f_c v=-g\frac{\partial z_s}{\partial x}-\frac{\tau_{b.x}}{\rho R}+\nu_t\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+f_c u=-g\frac{\partial z_s}{\partial y}-\frac{\tau_{b.y}}{\rho R}+\nu_t\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$

- Conservative Formulation
  - Version 6.0 and later

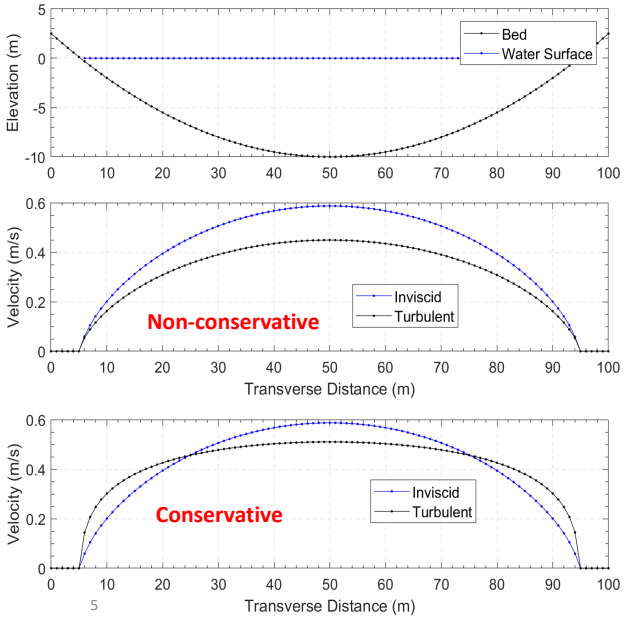
$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}-f_c v=-g\frac{\partial z_s}{\partial x}-\frac{\tau_{b.x}}{\rho R}+\frac{1}{h}\frac{\partial}{\partial x}\left(\nu_{t,xx}h\frac{\partial u}{\partial x}\right)+\frac{1}{h}\frac{\partial}{\partial y}\left(\nu_{t,yy}h\frac{\partial u}{\partial y}\right)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+f_c u=-g\frac{\partial z_s}{\partial y}-\frac{\tau_{b.y}}{\rho R}+\frac{1}{h}\frac{\partial}{\partial x}\left(\nu_{t,xx}h\frac{\partial v}{\partial x}\right)+\frac{1}{h}\frac{\partial}{\partial y}\left(\nu_{t,yy}h\frac{\partial v}{\partial y}\right)$$

4

- 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 for the increased dissipation.
- 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

- 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.
- This simple example clearly illustrates the problem with the non-conservative formulation.

Eddy Viscosity: Turbulence 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

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

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

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

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

6

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 turbulence 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 turbulence 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

Turbulence Numerical Implementations

- Non-conservative mixing
  - Eulerian-Lagrangian Solver

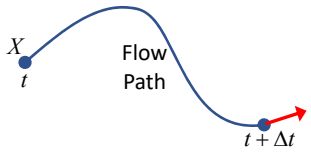
Eulerian-Lagrangian

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

- 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

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

- All mixing terms explicit
- Approximate Stability Criteria for Eulerian Solver

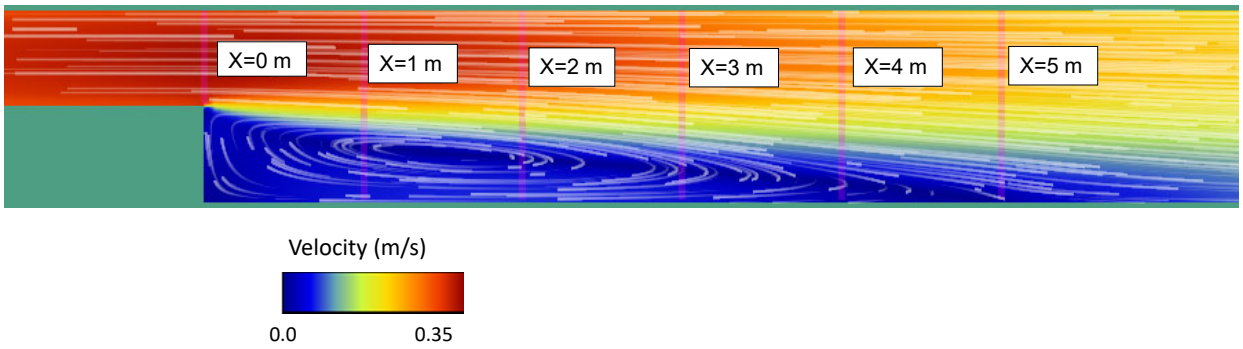
$$\frac{\nu_t\Delta t}{\Delta x^2}\le\frac{1}{2}$$

7

- 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.

Example: Sudden Expansion Lab Experiment

- Inflow: 0.039 m^3/s
- Downstream depth: 11 cm
- Slope: 0.0001
- Grid Resolution: 2.5 cm
- Time step: 0.02 and 0.0333 s
- Manning's n: 0.015 $s/m^{1/3}$



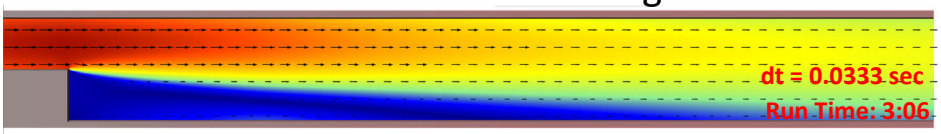
8

- OK, now I'm going to show some results for an example flume experiment of a sudden expansion.
- The purpose is compare the 2 flows solvers and mixing term formulations.
- The flow is from left to right and was run until steady state.
- The flow forms a recirculation zone after the sudden expansion and for that reason it's great test case for turbulence modeling.
- It also has good amount velocity measurements which were taken along 6 transects spaced 1-m apart starting the expansion.

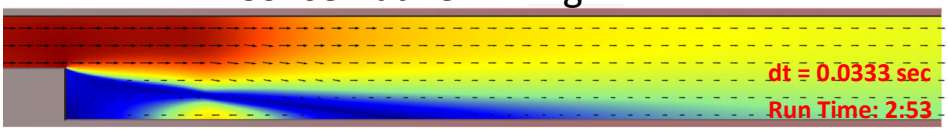
Results: Sudden Expansion Lab Experiment

ELM Solver

Non-conservative Mixing

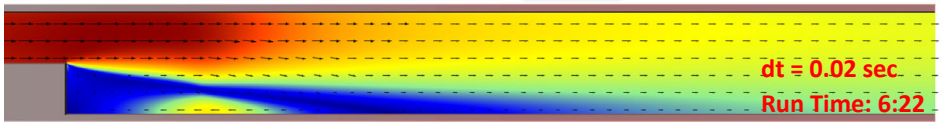


Conservative Mixing



EM Solver

Conservative Mixing



$D = 1.4$

$D_L = 0.6$

$D_T = 0.3$

$C_s = 0.1$

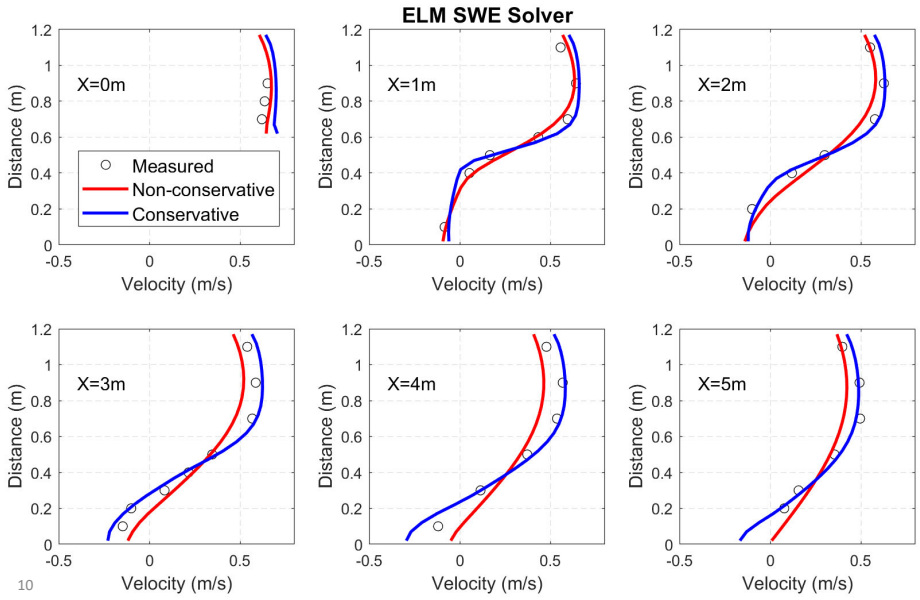
9

- This slide shows a comparison of the current velocity fields for the Sudden Expansion Lab Experiment computed with both the ELM and EM solvers and the two mixing term formulations.
- The top two results are the ELM solver with the non-conservative mixing term formulations, and the bottom result is the EM solver which only uses the conservative mixing formulation.
- The first thing notice is that the ELM and EM solvers produce essentially the same result with the turbulence setup which indicates that the flow results are not sensitive to the solver.
- However, comparing the 2 mixing term formulations, the results are very different. So different in fact that the mixing coefficients have to be calibrated independently. The calibrated values of the turbulence coefficients are shown on the right.
- The conservative mixing formulation produces a very weak return flow.
- For many lab scale test cases, the resolution can be so small that the stability criteria for the diffusion term of the EM solver can start to become important.
- This was precisely the case with test case and why the time step had to be reduced for the EM solver.
- The larger time step is one of the reasons the EM solvers was the slowest of the

three runs.

- Since the ELM solvers uses semi-Lagrangian and Lagrangian approaches for the mixing term, it doesn't have the same time step restriction as the EM solver and it can use a bigger time step.
- Of the three simulations, the ELM solver with the conservative mixing term was the fastest beating the non-conservative formulation just slightly.

Results: Sudden Expansion Lab Experiment



- This plot shows a comparison of measured and computed current velocities along the 6 transects of the sudden expansion experiment.
- In general the conservative mixing formulation does a better job in reproducing the velocities, especially the return flow which is most visible in the lower-left figure.

Turbulence Coefficients

- Best to calibrate coefficients with velocity measurements
- Non-conservative formulation generally requires larger values (2x) compared to the conservative formulation
- If coefficients are too large, the non-conservative formulation can produce excess dissipation and increase water levels
- When calibrating, start on the low end of the range

Mixing Intensity	Geometry and Surface	D_L	D_T
Weak	Straight channel Smooth Surface	0.1 to 0.3	0.04 to 0.1
Moderate	Gentle meanders Moderately irregular	0.3 to 1	0.1 to 0.3
Strong	Strong meanders Rough surface	1 to 3	0.3 to 1

Smagorinsky Coefficient: 0.05 to 0.2

11

- The table shows the recommended range for the longitudinal and transverse mixing coefficients as a function of different conditions.
- These are only approximate ranges and for the most accurate results it's best to calibrate the coefficients with measurements.
- As shown in the previous example, the non-conservative formulation generally requires mixing coefficients about twice as large as those with the conservative formulation.
- If the turbulence coefficient are too large the non-conservative formulation can produce excess dissipation and increase water levels.
- When calibrating, it's good to start the lower end of the recommended range.

Discussion and Conclusions

- Non-conservative formulation
  - Does not conservative momentum
  - Will generally lead to a net dissipation of momentum
  - Still an option in Version 6.0 for backwards compatibility
- Conservative formulation
  - Conserves momentum
  - Recommended approach for both EM and ELM solvers
  - Generally requires smaller mixing coefficients than the non-conservative formulation
  - EM Solver has a time step restriction
  - ELM solver does not have a time step restriction
  - EM and ELM solvers should produce similar results for most cases

12

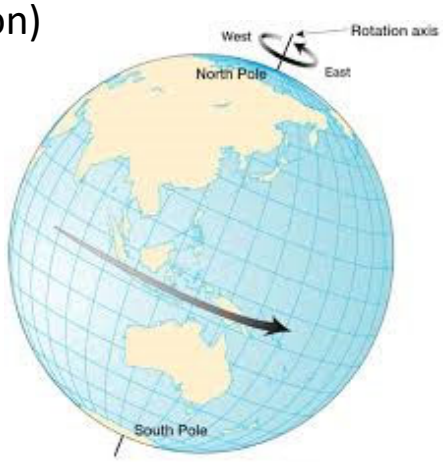
- The EM solver does have a time step limitation from by the mixing term.
- However, for most practical applications, the CFL condition is more restrictive.
- Currently, the variable time step algorithm in HEC-RAS does not check the diffusion stability criteria. This is something could be added in the future as an option.
- ELM solver does not have time step restriction based on mixing term because they are treated with a Lagrangian approach.
- And finally, the EM and ELM solvers should produce similar results for turbulence coefficients.

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



13

- 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.

Coriolis Numerical Implementations

- 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

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

f_c : Coriolis Parameter

θ : Implicit weighting factor

Δt : Time step interval

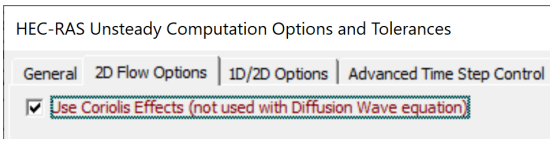
u_X^n : Backtracking velocity

14

- 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.

Coriolis Discussion and Conclusions

- Only for Shallow Water Equations (Not Diffusion-Wave Equation)
- Relatively inexpensive
- Does not impact stability or introduce time step limitations
- When in doubt, turn it on
- Requires one input parameter: average latitude for each 2D area
- Also requires checking a box



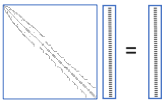
15

- The Coriolis acceleration is only applicable to the Shallow-Water Equations.
- In general, the Coriolis acceleration is relatively inexpensive to compute even for the Fractional Step Method so including doesn't add much computational time.
- In addition, the Coriolis acceleration does not introduce restrictive time step criteria or stability issues.
- Therefore, if there is any doubt that Coriolis should be included, it should be included.
- The best way to know if the Coriolis term is important, is to do a simple test of turning on and off and comparing the results.
- Turning off will have a minor impact on the computational speed.
- Including the Coriolis term only requires the average latitude for each 2D area. The Coriolis parameter is specific to each plan and 2D area.
- And finally if you want to include Coriolis, remember to check the box that says "Use Coriolis Effects" in the "Computation Options and Tolerances" window.

Matrix Solvers: Introduction

- Linear System of Equations

$$Ax = b$$



- Direct Solvers
  - Examples: Gaussian elimination, LU, Cholesky, and QR decompositions
  - Can be "black boxes"
  - Usually have few input parameters
  - High-accuracy
  - Can fail or be very slow for large matrices
  - Can be slow for unsteady or non-linear systems
- Iterative Solvers
  - Examples: GS, SOR, CG, GMRES
  - Require more options and input parameters
  - Usually require preconditioners, matrix balancing, ordering, etc.
  - Less accurate
  - Good for large problems
  - Good for unsteady or non-linear systems
  - Improper use can lead to instability problems or solution divergence

16

- All the 2D flow solvers in HEC-RAS require solving a sparse linear system of equations Ax = b.
- The method used to solve to for x are referred to simply as matrix solver.
- This is actually where most computational time is spent.
- And therefore, speeding up the matrix solver even by a little can significantly reduce the overall simulation run time.
- In general, there are 2 types of matrix solvers: direct and iterative.
- Examples of direct solvers are Gauss-Jordon elimination, LU, Cholesky, and QR decompositions
- Direct solvers can sometimes be "black boxes" in that they usually don't have many options and parameters and so users don't have to think too much about them.
- Direct solver are really accurate but can sometimes fail or be very slow for large, unsteady, or non-linear systems.
- Examples of iterative solvers are the Gauss-Seidel, Successive-Over-Relaxation, Conjugate Gradient, and Generalized Minimal RESidual solvers.
- Iterative solvers usually have more options and input parameters
- And usually, to be effective, iterative solvers need additional methods such as preconditioning, matrix balancing, ordering, etc.

- And these methods may have their own options and input parameters.
- For these reasons, iterative solver require more user skill and tuning for best results.

HEC-RAS Direct Solver

- PARDISO

  High-performance, robust, memory efficient and easy-to-use solver for symmetric and non-symmetric linear systems.
  - Version in Intel Math Kernel Library
  - Parallel on PC's
  - Can be used as a "black box"
    - Very little parameters
    - No need matrix balancing, ordering, etc.

17

- HEC-RAS supports one direct matrix solver called PARDISO which is a high-performance, and robust solver.
- The version in RAS is distributed with the Intel Math Kernel Library and based on a version PARDISO from 2006.
- It uses multiple cores, runs very efficiently, and easy to use
- It can be used almost as a "black box" because
- As long as you know a few things about the matrix and right-hand-side, it's pretty easy to use.
- It doesn't require matrix, ordering, preconditioning, or anything like that.
- It handles all of that internally, and automatically.

HEC-RAS Iterative Solvers

- SOR: Successive Over-Relaxation
  - Relaxation factor ($0 < \omega < 2$)
  - Asynchronous (ASOR) parallel implementation
  - Extremely simple
  - May take many iterations to converge but each iteration is very inexpensive
- FGMRES-SOR: Flexible Generalized Minimal RESidual
  - "Flexible" variant of GMRES which allows preconditioner to vary from iteration to iteration
  - SOR as preconditioner

18

- HEC-RAS supports 2 iterative matrix solvers: SOR and FGMRES-SOR
- SOR stands for successive over-relaxation
- The solver is a variation of the Gauss-Seidel solver but with an over-relaxation factor that speeds up convergence.
- The parallel implementation in RAS is called Asynchronous parallel SOR (ASOR).
- The solver and code are extremely simple
- So even though it may take many iterations to converge than others solvers, it can still faster because it each iteration is very fast.
- The second option is the FGMRES-SOR solver.
- FGMRES stands for Flexible Generalized Minimal Residual.
- This flexible variant of GMRES allows for the preconditioner to vary from iteration to iteration.
- We tried several preconditioners including some incomplete Lower-Upper preconditions and SOR tended to be the fastest.

Iterative Solver Input Parameters

- Convergence Tolerance
  - Determines the overall accuracy
- Minimum Iterations
  - Increases accuracy, avoids solution drift, and allows the solver to stabilize
- Maximum Iterations
  - Avoids stalling and too many iterations caused by a small convergence tolerance
- Restart Iteration (Only FGMRES-SOR)
  - Reduces run time and memory requirements
- Relaxation Factor
  - Used for both SOR solver and preconditioner
- SOR Preconditioner Iterations (Only FGMRES-SOR)
  - In-lieu of checking convergence which would be slower

Parameter	Range
Convergence Tolerance	0.001 – 0.000001
Minimum Iterations	3 – 6
Maximum Iterations	5 – 30
Restart Iteration (FGMRES Only)	8 – 12
Relaxation Factor	1.1 – 1.5
SOR Preconditioner Iterations (FGMRES Only)	5 – 20

19

- Depending on the iterative matrix the user may need to specify either 4 or 6 input parameters.
- The SOR solver has 4 input parameters and the FGMRES solver has an additional 2.
- The Convergence Tolerance determines the overall accuracy the of the solver.
- The minimum iterations increases accuracy, avoids solution drift, and allows the solver stabilize.
- The maximum iterations avoids stalling and computing too many iterations when the convergence tolerance is too small.
- The restart iteration is only applicable the FGMRES solver. It's not a very sensitive parameter and can usually be left at around 10.
- The relaxation factor is used by both the SOR solver and preconditioner.
- SOR Preconditioner iterations obviously only applies to FGMRES solver.
- A constant number of iterations is used instead of checking for convergence because it's faster and simpler.

Iterative Solvers: Stopping Criteria

- Error Estimate

$$E^m = \frac{\left\| \boldsymbol{D}^{-1}\left(\boldsymbol{A}\boldsymbol{x}^m - \boldsymbol{b}\right)\right\|_2}{\sqrt{N}}$$

$\boldsymbol{D}$: Diagonal of $\boldsymbol{A}$

$\boldsymbol{A}$: Coefficient matrix

$\boldsymbol{x}$: Solution

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

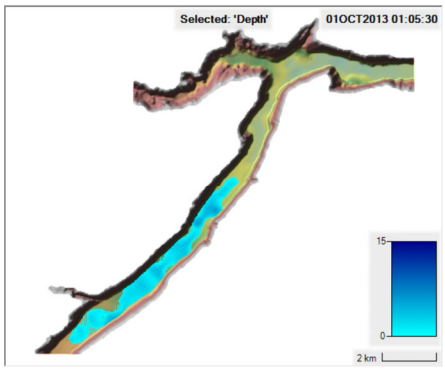
N : Number of rows

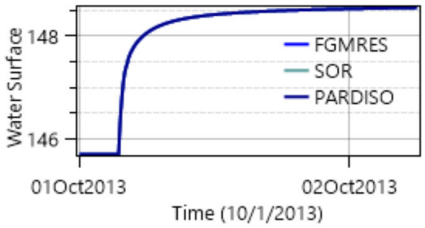
Iterative Solver Status	Criteria	Description
Iterating	$N_{\min} < m < N_{\max}$ and $E^m > T_C$ and $\frac{E^{m-1} - E^m}{E^1 - E^m} > T_S$ and $E^m < E^1$	Iterative solver is converging and will continue to iterate. N_{min}: Minimum number of iterations N_{max}: Maximum number of iterations T_C : Convergence tolerance T_S = 0.1T_C : Stalling tolerance
Converged	$E^m \le T_C$	Convergence criteria met. Solution accepted.
Stalled	$\frac{E^{m-1} - E^m}{E^1 - E^m} \le T_S$	Convergence rate has decreased to an insignificant level without satisfying the converged criteria. The current solution is accepted and the iteration loop is exited.
Max Iterations	$m = N_{\max}$	Maximum number of iterations reached without reaching the converged criteria. The current solution is accepted and the iteration loop is exited.
Divergent	$E^m > E^1$	Iterative solution is divergent. Either the normalized residuals are getting larger, or a Not a Number (NaN) has been detected.

20

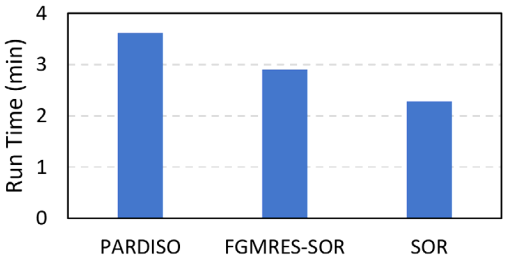
- There are 4 stopping criteria for the iterative solvers
- These determine if the solver should continue iterating, or if it stop because it either converged, stalled, diverged, reached the maximum number of iterations.
- The equation for the error estimate is shown on the left. Note the diagonal scaling of the residuals.
- The error is basically a root-mean-squared of the normalized residuals.
- For simplicity, the convergence criteria does not include the maximum of the residuals.

Example: EA Test 5 (Dam Failure)





Setting	FGMRES-SOR	SOR
Convergence Tolerance	0.0001	0.0001
Minimum Iterations	3	5
Maximum Iterations	20	30
Restart Iteration	10	
Relaxation Factor	1.3	1.3
SOR Preconditioner Iterations	10	



21

- UK Environmental Agency released a report in 2013 with a benchmark test cases for 2D hydraulic models.
- This example is test No. 5 of that report.
- It's a flooding simulating of a river valley about 17-km long and 1-kg wide.
- The animation on the top left shows the flooding simulation.
- The test case was run in RAS with 3 available matrix solvers.
- The table on the right shows the parameters of the iterative solvers.
- The bar plot on the lower right shows the run times with the 3 solvers.
- Compared PARISO the FGMRES, and SOR solvers has speed-up factors of 125%, and 160% respectively.
- These were all run with 8 cores.
- The plot on the lower left shows a comparison of water levels at a point downstream and demonstrates that the 3 solvers produced essentially the same result.
- The iterative solver parameters not calibrated but simply estimated based on the recommended range and experience.
- So they could be optimized to get even faster run times.

Matrix Solver Best Practices

- Start with PARDISO
- Ensure model is stable and not going to max iterations every time step or reporting large water surface or volume errors
- Try FGMRES and SOR solvers
- Start with conservative parameters
- Compare with PARDISO
- Adjust parameters to optimize run time
- Iteration parameters left empty or set to zero are assigned a default value based on mesh size

22

- When starting a model, it's recommended to start with PARDISO.
- This is so that you have a reference to compare the iterative solver results and run times.
- Ensure the model is stable and running well.
- This means it's not going to max iterations every time step and reporting large water surface or volume errors.
- Once the model is running well, try the iterative solvers starting with FGMRES.
- Start with conservative parameters, this means a small convergence tolerance and a lot of iterations.
- Compare the iterative solver results to PARDISO to make there is no solution drift.
- If there is, then the convergence tolerance may be too large.
- The maximum number of iterations may be too large.
- Or the minimum number of iterations may be too small.
- Adjust the parameters to optimize the run time.
- Optimizing the iterative solver parameters can sometimes mean finding a balance between accuracy and computational time.
- This means increasing the convergence tolerance, reducing the number of iterations and increasing the relaxation factor.
- One useful tip is that any iteration parameters left empty or set to zero is

automatically assigned a default values based on the mesh size.

- This can be used to get an initial estimate for different mesh sizes.

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

23