Turbulence Modeling

What Every CFD Engineer Should Understand

Lesson 6 from 38 Years of Teaching CFD

k-epsilon • k-omega • SST • RSM • Wall Treatment

Turbulence Modeling: What Every CFD Engineer Should Understand

Lesson 6 from 38 Years of Teaching CFD

Introduction

Turbulence modeling is often treated as a black box -- just pick k-epsilon or k-omega, hit 'Run,' and hope for the best.

But behind those dropdown menus lie decades of theory, assumptions, and trade-offs. There is no analytical solution to even the simplest turbulent flow -- such as fully developed channel flow. While Direct Numerical Simulation (DNS) is possible in theory, it's computationally impractical for most real-world flows.

That's where turbulence models come in: they allow us to predict the mean behavior of turbulent flows at a fraction of the cost.

Why Turbulence Modeling Exists

In practical engineering -- from wind flow around buildings to combustion in engines -- we're interested in mean quantities like drag, lift, or wall heat flux.

To compute those, we use the Reynolds-Averaged Navier--Stokes (RANS) equations, which average the effects of turbulence. But doing so introduces new unknowns -- the Reynolds stresses -- without adding new equations.

That's the famous closure problem. We resolve it by introducing turbulence models -- mathematical expressions for the turbulent viscosity or Reynolds stresses.

Core Concepts Every Engineer Must Know

- Turbulent Kinetic Energy (k): Represents the energy in turbulent velocity fluctuations.
- Eddy Viscosity Assumption: Modeled as nu_t = C_mu k^2/epsilon (in k-epsilon), or nu_t = k/omega (in k-omega).
- Kolmogorov's Legacy: Foundation of two-equation modeling.
- Reynolds Stress Models (RSM): More detailed but costly.

Turbulence Modeling: What Every CFD Engineer Should Understand

Lesson 6 from 38 Years of Teaching CFD

- Isotropy Assumption: Limits accuracy in anisotropic flows.

When Popular Models Work -- and When They Don't

Spalart--Allmaras: Fast, simple, great for external flows | Poor for separation

k-epsilon: Robust, widely validated | Inaccurate near walls / separation

k-omega SST: Good near-wall + free-stream | Sensitive to freestream values

RSM: Captures anisotropy | Expensive, complex to converge

Common Mistakes Engineers Make

- Ignoring wall treatment: Most errors stem from wall functions.
- Blind use of defaults: Always review solver setup.
- Model-blindness: Understand your model's domain of validity.

My Teaching Experience

In class, I assign channel flow prediction using both k-epsilon and k-omega. Students are asked to:

- Vary constants in the k-epsilon model
- Examine wall treatment effects

The lesson: small changes can lead to significant differences.

Key Takeaways

- Turbulence models are not interchangeable
- Understand assumptions before applying
- Wall treatment is critical
- Always validate -- modeling is part science, part art

Looking Ahead

Turbulence Modeling: What Every CFD Engineer Should Understand

Lesson 6 from 38 Years of Teaching CFD

In Lesson 7, I'll discuss validation: What counts as validation -- and how do you know your CFD results are trustworthy?

CFD Validation

How Do You Know Your Simulation Is Right?

Lesson 7 from 38 Years of Teaching CFD

Verification • Validation • Uncertainty
Best Practices for Trustworthy Simulations

CFD Validation: How Do You Know Your Simulation Is Right?

Lesson 7 from 38 Years of Teaching CFD

Introduction

In this lesson, I discuss Verification and Validation (V&V) in CFD.

V&V is a huge subject, but here I focus on:

- What V&V really means
- A hierarchical method for validation
- Strategies for quantifying uncertainty and errors
- Recommended best practices for credible simulations

Key Concepts and Definitions

Verification: Ensuring the computational model correctly solves the equations (code accuracy).

"Are we solving the equations right?"

Validation: Determining how well the model represents physical reality.

"Are we solving the right equations?"

Uncertainty: Lack of knowledge due to unknown input parameters or model form.

Error: Recognized deficiency not due to lack of knowledge (e.g., coding or discretization errors).

Verification: Theory and Practice

Goal: Minimize and quantify numerical errors in the code.

Strategies:

- Compare code output to analytical or high-accuracy numerical solutions
- Check stability and convergence systematically
- Address the five primary sources of numerical error:
  1. Spatial/temporal discretization
  2. Iterative convergence errors
  3. Round-off errors

4. Programming errors
5. Model implementation mistakes

Software Quality Engineering (SQE) ensures the solver is consistent, testable, and traceable.

Validation: Strategy and Methodology

Goal: Determine how well predictions agree with experiments.

Types:

- Model Validation -> Checks the mathematical formulation
- Solution Validation -> Checks a specific case or setup

Hierarchical Strategy:

1. Start with simple problems (pipe flow, flat plate)
2. Progress to complex, integrated systems (e.g., missile flight, heat exchangers)
3. Compare global quantities first (drag, pressure drop, Nusselt number)
4. Use experiments designed for validation, not just general research data

Experimental Guidelines:

- Define boundary & initial conditions precisely
- Estimate measurement uncertainty
- Repeat experiments whenever possible

Quantifying Uncertainty

1. Characterize experimental uncertainty
2. Incorporate input variability in CFD simulations
3. Use nondeterministic ensemble simulations to produce a range of outputs

Tip: Distinguish between

CFD Validation: How Do You Know Your Simulation Is Right?

Lesson 7 from 38 Years of Teaching CFD

- Aleatory uncertainty -> inherent randomness
- Epistemic uncertainty -> lack of knowledge

Validation Metrics and Comparison

- Use confidence intervals and visual comparisons
- Quantify agreement, but remember:

"Agreement != Model Validity" -- it only supports credibility within defined limits

Key reminder: Validation is not about exact matches -- it's about building trust in your model.

My Teaching Approach

I assign projects where students:

- Compare CFD with benchmark cases (pipe flow, flat plate)
- Explain why models may differ from experiments

Core Message:

"A good simulation doesn't just give the right number -- it gives confidence it will keep doing so under different conditions."

Key Takeaways

- Validation compares simulation to physical reality
- Understand your data and assumptions
- A mismatch is a chance to learn, not failure
- A validated model has predictive power

Your Experience?

How do you approach CFD validation in your work?

What's the toughest validation challenge you've faced?

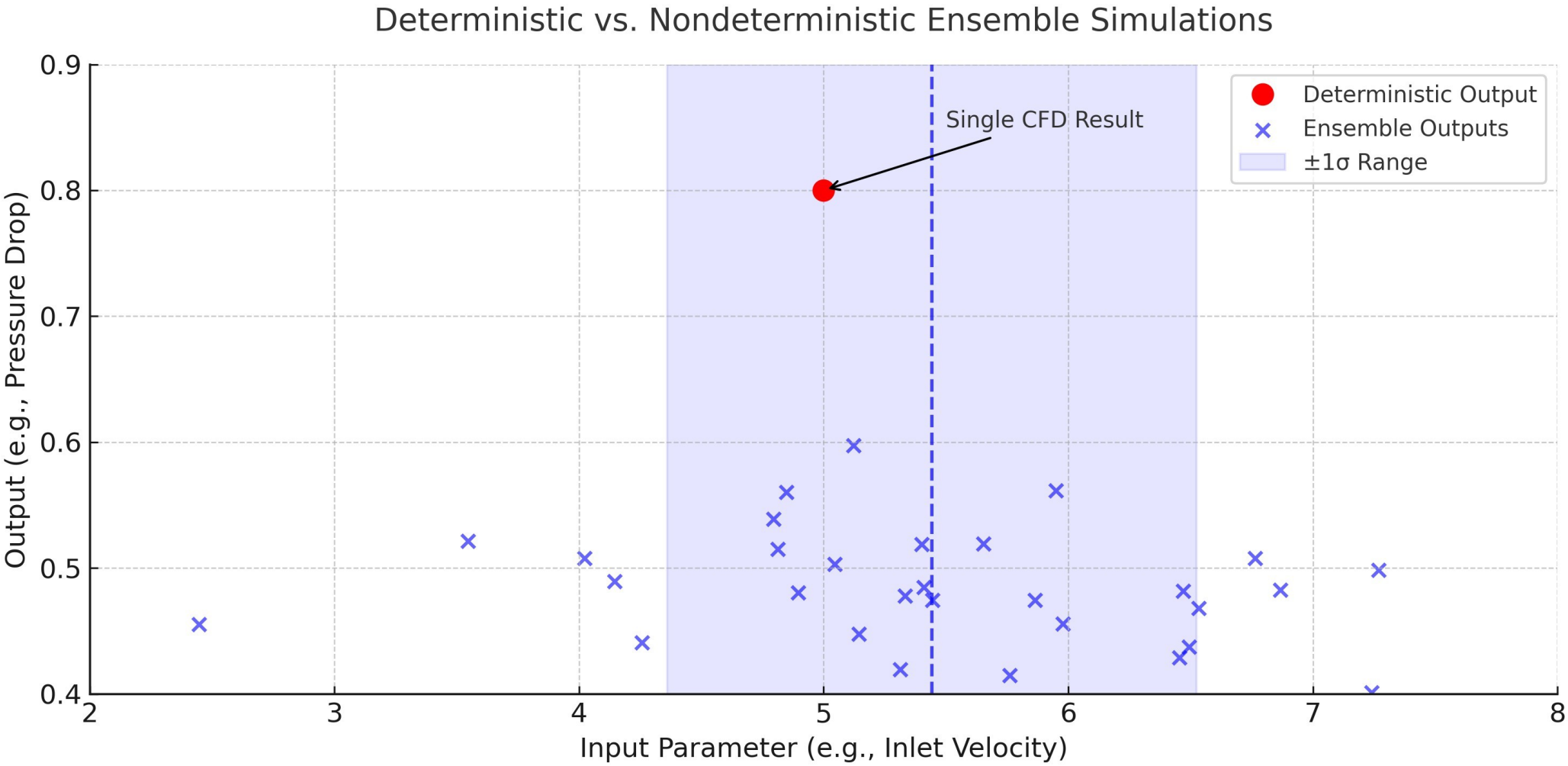
CFD Validation: How Do You Know Your Simulation Is Right?

Lesson 7 from 38 Years of Teaching CFD

Share your experience -- let's learn together!

Coming in Lesson 8

I'll share a CFD modeling challenge that tested my patience and persistence -- and what it taught me about the limits of automation.


Deterministic vs. Nondeterministic Ensemble Simulations
Deterministic Output
Ensemble Outputs
±1σ Range
Single CFD Result
Output (e.g., Pressure Drop)
Input Parameter (e.g., Inlet Velocity)
0.9
0.8
0.7
0.6
0.5
0.4
2
3
4
5
6
7
8

CFD Challenges

Patience, Persistence, and Debugging Lessons

Lesson 8 from 38 Years of Teaching CFD

Multigrid Stories • Debugging • Solver Choices

CFD Challenges That Tested My Patience and Persistence

Lesson 8 from 38 Years of Teaching CFD

Background

About 35 years ago, I was implementing a Multigrid (MG) algorithm for a 2D heat conduction problem.

- Computational domain: a square grid
- Same number of points in X and Y directions
- Base solver: Gauss-Seidel (GS) iteration
- Multigrid: Used to accelerate convergence on finer grids

Why Multigrid?

* Speeds up simple iterative solvers like GS
* In theory, reduces or removes the need for a full grid-independence study

I set up five grid levels: 8x8 -> 16x16 -> 32x32 -> 64x64 -> 128x128

Multigrid Workflow (Simplified)

1. Perform a few GS iterations on the coarsest grid (8x8) to reduce residuals.
2. Prolong the solution and residual to the next finer grid (16x16) and iterate to reduce residuals by ~1 order of magnitude.
3. Continue prolongation up through 32x32 -> 64x64 -> 128x128 grids.
4. Restrict the residuals and solution values from fine -> coarse grids and perform iterations again.
5. Repeat V-cycles until the normalized residual drops below 1e-4 on the finest grid.

The Problem

The solver converged on the 128x128 grid... but there was NO Multigrid acceleration.

Weeks of debugging followed. Finally, we discovered the culprit:

In the restriction routine, a single line had a typo:

Instead of multiplying by 0.25, it was 0.5!

CFD Challenges That Tested My Patience and Persistence

Lesson 8 from 38 Years of Teaching CFD

Once fixed, Multigrid acceleration was night and day.

Conclusion and Reflection

While fixing that one line was satisfying, I left with two lessons:

1. Multigrid demands perfection - even small coding errors kill performance.

2. Simplicity wins in practice - we switched to a preconditioned conjugate gradient solver and never looked back.

Lesson Learned: Advanced algorithms are powerful, but robustness and simplicity often win in the long run.

Key Takeaways

* Debugging CFD requires patience and systematic thinking
* Multigrid is sensitive to small coding errors
* Sometimes, simpler solvers offer more reliable performance

Your Experience?

Have you ever spent weeks chasing a single line of code in a CFD project?

Share your most frustrating (or rewarding) debugging story!

CFD Challenges That Tested My Patience and Persistence

Lesson 8 from 38 Years of Teaching CFD

Multigrid V-Cycle Illustration