Chapter 15

Python Code

15.1 Companion Chapter 3: Scalar Mixing

Interactive Companion for Chapter 3: Scalar Mixing

This Jupyter notebook accompanies Chapter 3 and lets students *experiment* with the core closures and balances introduced in the text. It blends short theory reminders with widgets, simple solvers, and practice tasks so learners can vary parameters and immediately see how turbulent transport responds.

Learning goals

- Connect the mixing–length hypothesis to eddy viscosity/diffusivity and scalar flux.
- Visualize how near–wall damping (Van Driest) alters ℓ_m, ν_t, and wall gradients.
- Solve steady 1D diffusion–form mean equations with molecular + turbulent transport.
- Interpret stability via N^2 and Ri_g, and relate it to scalar mixing.
- Work through practical scenarios (HVAC jet, river outfall, stack plume, density current).

What's inside (structure)

0. Core equations (quick reference). The notebook opens with the main relations used throughout the chapter:

$$\nu_t = \ell_m^2 \left|\frac{\partial U}{\partial y}\right|, \qquad \kappa_t = \frac{\nu_t}{Pr_t}, \qquad q_y = -\,\kappa_t\,\overline{T}_y, \qquad P_\theta = 2\,\kappa_t\,\overline{T}_y^2.$$

and the 1D diffusion–form balances (molecular + turbulent):

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial y}\Big[(\nu + \nu_t)\,U_y\Big], \qquad \frac{\partial \overline{T}}{\partial t} = \frac{\partial}{\partial y}\Big[(\alpha + \kappa_t)\,\overline{T}_y\Big].$$

A. Shear–driven mixing (mixing length). A short primer recaps ℓ_m as a length scale and uses a shear–based velocity scale to recover $\nu_t = \ell_m^2|\partial U/\partial y|$ and $\kappa_t = \nu_t/Pr_t$. An interactive widget lets students choose ℓ_m forms (linear, with Van Driest damping), set Pr_t, pick a shear profile, and inspect the resulting ν_t, κ_t, q_y, and P_θ profiles.

A.1 Theory: Linear ℓ_m and Van Driest damping. A compact theory block introduces wall units $y^+ = y\,u_\tau/\nu$ and the near–wall damping

$$\ell_m(y) = \kappa\,y\Big[1 - \exp\big(-\,y^+/A^+\big)\Big]^m, \qquad y^+ = \frac{y\,u_\tau}{\nu},$$

with typical parameters $\kappa \simeq 0.41$, $A^+ \simeq 26$, $m \in \{1,2\}$. A small widget compares $\ell_m = \kappa y$ vs. damped ℓ_m and the induced ν_t.

B. Steady diffusion–form solvers (1D). A minimal finite–difference solver demonstrates steady solutions of

$$\frac{d}{dy}\Big[(\nu + \nu_t)\,U_y\Big] = 0, \qquad \frac{d}{dy}\Big[(\alpha + \kappa_t)\,\overline{T}_y\Big] = 0,$$

with Dirichlet boundary conditions, iterating the nonlinear coupling $\nu_t(\,U_y\,)$.

C. Buoyancy effects: N^2 and Ri_g. A brief recap of Boussinesq stability measures (N^2 and $Ri_g = N^2/(U_y)^2$) is paired with a widget showing how stability and Pr_t influence κ_t and, by implication, fluxes. Stable regions (large Ri_g) suppress vertical mixing.

D. Density current (gravity current). An interactive bulk model uses reduced gravity $g' = g\,\Delta\rho/\rho_0$, a front Froude number Fr, and a simple entrainment coefficient E to evolve thickness $h(x)$ and dilution $\Delta C(x)$. A mixing–length proxy ($\ell_m \sim \gamma h$) provides illustrative κ_t and a scalar–flux proxy $q(x)$.

E. Synthetic shear data examples. Recreates the synthetic profiles from the shear companion: (i) clean toy profiles, (ii) noisy data with CSV export, and (iii) reconstruction checks comparing ν_t, κ_t recomputed from (ℓ_m, U_y, Pr_t) against stored columns—useful for unit tests and plotting practice.

F. Self–assessment. Short multiple–choice and brief–answer questions on mixing length, Pr_t, stability effects, and diffusion–form balances (answers are provided in collapsible form inside the notebook).

G. Homework (practical). Three open–ended assignments with starter code:

- HVAC ceiling jet near a wall: compute $\nu_t, \kappa_t, q_y, P_\theta$; solve steady momentum with $\nu + \nu_t(y)$.
- River outfall (weak stratification): map $Ri_g(y)$ and compare fluxes as Pr_t varies.
- Stack plume: stable/neutral/unstable N^2 scenarios and the impact on q_y.

How to use it

1. Launch Jupyter (Notebook or Lab) in the folder containing the notebook. Ensure MathJax (for equations) and `ipywidgets` (for sliders) are enabled.
2. Run the “Setup” cell; then explore the widgets in Sections A, C, and D.

3. Optional CSVs (if available in the same folder): `Chapter3Companion_ScalarMixing_Shear.csv`, `Chapter3Companion_S`
`Chapter3Companion_ScalarMixing_Buoyancy_Unstable.csv`.

Files

- **Chapter3_Mixing_Interactive.ipynb** — main interactive notebook (this companion).
- **ch3_scalar_mixing.py** — tiny helper module (mixing length, ν_t, κ_t, loaders).

Outcomes

By the end, students can (i) construct ν_t, κ_t from ℓ_m, U_y, Pr_t, (ii) reason about how N^2 and Ri_g modulate scalar mixing, (iii) solve simple steady diffusion–form problems, and (iv) interpret practical scenarios by linking parameters to observable changes in fluxes and production.

15.2 Chapter 4:Turbulence Equations: Python Coding for Mechanical and Buoyant Turbulent Production

The full scripts are available in the `GitHub` repository for this book (`https://github.com/CFD-UTSA/UnsteadyBookBhaganagarr/blob/main/Chapter2-Mixing/README.md`) Interactive Jupyter notebooks are also provided so that readers can run, modify, and visualize results **Note on filenames.** This appendix follows the

repository names `mechanical_turbulence_timescales.py`, `mixing_timescal`
`mixing_workflow.py`, and `buoyancy_turbulence_timescales.py`.

A. Mechanical Turbulence (3 files)

Files.

- `mechanical_turbulence_timescales.py` (case generator; produces $k(t)$, $\varepsilon(t)$, etc.)

- `mixing_timescales.py` (utility for τ_s, τ_e, η, τ_η, u_η, λ, etc.)
- `mixing_workflow.py` (end-to-end driver: run a mechanical case $\to$ compute/print time scales)

Common notation and closures.

- TKE: $k = \frac{1}{2}\overline{u_i' u_i'}$, dissipation ε; mean shear $S = \partial U/\partial z$.
- Production: $P = -\overline{u'w'}\, S \approx \nu_t S^2$, with eddy viscosity $\nu_t \approx \ell_m^2 \,|S|$.
- Dissipation model: $\varepsilon \approx C_\varepsilon \, k^{3/2}/\ell_\varepsilon$ (often $\ell_\varepsilon \sim \ell_m$).
- Time/length micro-scales: $\eta = (\nu^3/\varepsilon)^{1/4}$, $\tau_\eta = (\nu/\varepsilon)^{1/2}$, $u_\eta = (\nu\varepsilon)^{1/4}$.
- Clocks: $\tau_s = 1/|S|$, $u' \approx \sqrt{2k/3}$, $\tau_e \approx L/u'$ (eddy turnover).

Cases included in `mechanical_turbulence_timescales.py`.

1. **Shear-production box** (`box`):

$$\frac{dk}{dt} = P - \varepsilon, \quad P \approx \nu_t S^2, \quad \nu_t \approx \ell_m^2 |S|.$$

Inputs: S, ℓ_m, C_ε, k_0, t_{end}, Δt. *Outputs:* $k(t)$, $\varepsilon(t)$, τ_s, τ_e, $(\eta, \tau_\eta, u_\eta)$.

```
python mechanical_turbulence_timescales.py box --S 3.
```

2. **Plane Couette** (`couette`): steady 1D momentum with $\nu_{\text{eff}} = \nu + \nu_t$,

$$\frac{d}{dz}\left(\nu_{\text{eff}} \frac{dU}{dz}\right) = 0 \;\Rightarrow\; \nu_{\text{eff}} \frac{dU}{dz} = \tau_w.$$

Inputs: H, U_0, ν_t or mixing-length params, grid. *Outputs:* $U(z)$, representative S_{rep}, optional k, ε.

```
python mechanical_turbulence_timescales.py couette -
```

3. **Grid-generated decay** (`grid`):

$$\frac{dk}{dt} = -C_\varepsilon \frac{k^{3/2}}{\ell} \quad \Rightarrow \quad k(t) \sim (t+t_0)^{-2}.$$

Inputs: k_0, ℓ, C_ε, t_{end}, Δt. *Outputs:* $k(t)$, $\varepsilon(t)$, decay time $\tau \sim \ell/\sqrt{k}$.

```
python mechanical_turbulence_timescales.py grid --k0
```

4. **Oscillatory shear** (`osc`): $S(t) = S_0 \sin(\omega t)$,

$$\frac{dk}{dt} = P(t) - \varepsilon, \quad P(t) \approx \nu_t S(t)^2, \quad \nu_t \approx \ell_m^2 |S(t)|.$$

Inputs: S_0, ω, ℓ_m, C_ε, k_0, t_{end}, Δt. *Outputs:* $k(t)$, $\varepsilon(t)$, phase/amplitude vs. $S(t)$.

```
python mechanical_turbulence_timescales.py osc --S0 4
```

Computing canonical scales (mechanical). Use `mixing_timescales.py` directly, or `mixing_workflow.py` to automate.

```
# Direct: supply S, L, k, eps, nu
python mixing_timescales.py --S 3.0 --L 0.1 --k 0.2 --eps

# One-click workflow (runs a case -> prints scales)
python mixing_workflow.py
python mixing_workflow.py couette --save
```

B. Buoyancy-Generated Turbulence (1 file)

File.

- `buoyancy_turbulence_timescales.py` (convective and free-fall scalings)

Cases included.

1. **Convective mixed layer (Deardorff).**

$$w_* = (B\,H)^{1/3}, \qquad \tau_b = H/w_*.$$

Inputs: B (m^2/s^3), H (m). *Outputs:* w_* (m/s), τ_b (s).

```
python buoyancy_turbulence_timescales.py —B 0.03 —H
```

2. **Free-fall (Rayleigh-Bénard-like).**

$$u_{ff} = \sqrt{g\,\beta\,\Delta T\,H}, \qquad \tau_{ff} = H/u_{ff}.$$

Inputs: g (m/s^2), β (K^{-1}), ΔT (K), H (m). *Outputs:* u_{ff} (m/s), τ_{ff} (s).

```
python buoyancy_turbulence_timescales.py —g 9.81 —b
```

C. Quick Crosswalk (Mechanical ↔ Scales ↔ Buoyancy)

Generator	Primary inputs	Primary outputs / scales
Shear box / Couette / Osc.	S, ℓ_m, k_0, model consts	$k(t)$, $\varepsilon(t)$; τ_s, τ_e, η, τ_η, u_η, λ
Grid decay	k_0, ℓ, C_ε	Decay of $k(t)$, $\varepsilon(t)$; $\tau \sim \ell/\sqrt{k}$
Convective CBL (Deardorff)	B, H	$w_* = (BH)^{1/3}$, $\tau_b = H/w_*$
Free-fall (R–B–like)	g, β, ΔT, H	$u_{ff} = \sqrt{g\beta\Delta T H}$, $\tau_{ff} = H/u_{ff}$

15.3 Companion Chapter 4: Turbulence Equations

Computational Notebooks for Chapter 4

RANS_Basics_ChapterCompanion.ipynb. Prepares the ground for Chapter 4 by reviewing Reynolds decomposition, av-