NONLINEAR RADIATION CAPSULE, VERSION 1.0

NOAH PETTINATO, CODE AND DOCUMENTATION EDITOR
MALGORZATA PESZYNSKA

1. INTRODUCTION

In this document, we present a MATLAB-based study of the one-dimensional (1D) heat equation with nonlinear radiation boundary conditions. The project addresses steady-state and time-dependent cases with Dirichlet and Robin-type boundaries, using a nonlinear flux model of the form $q = \alpha(u^4 - u_\star^4) + g$. Spatial discretization is performed using finite difference (FD) and centered control finite difference (CCFD) methods, with ghost points in the latter for improved boundary accuracy. Time integration uses the fully implicit backward Euler scheme. Nonlinear boundary terms are treated via Newton's method with analytic Jacobian assembly. The study proceeds through five phases of increasing complexity, culminating in a fully implicit solver for time-dependent radiation problems. We test solver stability and convergence across varying parameters, including stiff nonlinearities and large boundary coefficients.

1.1 **Acknowledgements.** This research was supported by the Oregon State University Faculty Research Assistant Program. I would like to thank Professor Malgorzata Peszynska for her mentorship and guidance throughout this project.

1.2 **Version.** This README document corresponds to version 1.0 of the MATLAB capsule. It describes the project structure and includes instructions for running each script by phase.

1.3 **Contents.** The current version of this capsule contains the following files, functions, and sub-folders:

```
Project-Folder/
    README.pdf
    Phase0/
        robin.m
    Phase1/
        animate_convergence_epsilons.m
        compare_convergence_epsilons.m
        convergence_plot.m
        f_and_J.m
        newton_eq20.m
    Phase2/
        f_and_J_robin_ccfd.m
        f_and_J_robin_fd.m
        newton_eq20.m
        robin_phase2.m
        robin_phase2_convergence.m
    Phase3/
        phase3_ccfd_step.m
        phase3_fd_step.m
        robin_phase3.m
    Phase4/
        f_and_J_phase4_ccfd.m
        f_and_J_phase4_fd.m
```

Date: Aug. 17, 2025.

```
newton_eq20.m
phase4_ccfd.m
phase4_fd.m
phase4_compare.m
```

Further descriptions may be found in the following Sections of this document:

- **Model Description**; see Section 2
  We describe the governing PDEs, boundary and initial conditions used across phases.
- **Numerical Methods**; see Section 3
  We summarize the FD and CCFD spatial schemes, time discretizations, and the Newton-based approach for nonlinear boundary flux.
- **Nonlinear Solver Implementation**; see Section 4
  We detail Newton's method, the modular boundary flux routines, and analytic Jacobian structure.
- **MATLAB Code Structure**; see Section 5
  We outline the modular scripts by phase and provide guidance on running simulations.
- **Results and Observations**; see Section 6
  We present numerical results from each phase and validate solver behavior.
- **Supplementary Information**; see Appendix A and B
  Includes figures, convergence plots, and MATLAB Scipts.

2. Model Description

2.1 **Notation.** We consider one-dimensional heat conduction on $x \in (0, 1)$ and $t \in (0, T]$, with temperature $u(x, t)$ governed by a diffusion equation and radiative boundary flux. Temperature is in Kelvin (K), time in seconds (s). The domain length is nondimensionalized to unity.

Table 1: Primary variables and parameters.

Symbol	Description	Unit
Variables		
$u(x, t)$	Temperature	K
x	Spatial coordinate	–
t	Time	s
Parameters		
k	Thermal conductivity	$\mathrm{W\,m^{-1}\,K^{-1}}$
c	Volumetric heat capacity	$\mathrm{J\,m^{-3}\,K^{-1}}$
$f(x)$	Internal source	$\mathrm{W\,m^{-3}}$
α	Radiation coefficient	$\mathrm{W\,m^{-2}\,K^{-4}}$
$u_\star$	Reference temperature	K
g	External heat flux	$\mathrm{W\,m^{-2}}$

2.2 **Governing Equations.** The model PDE is

$$cu_t = ku_{xx} + f(x), \tag{2.2a}$$

with steady-state given by $u_t = 0$. Initial condition for time-dependent cases (Phases 3–4):

$$u(x, 0) = u_{\text{init}}(x). \tag{2.2b}$$

Boundary conditions:

$$u(0,t) = 0 \quad \text{(Dirichlet)}, \tag{2.2c}$$

$$-ku_x(1,t) = q, \quad q = \alpha(u^4 - u_\star^4) + g. \tag{2.2d}$$

Pollock Nonlinear System (Phase 1). To test Newton's method in isolation from PDE discretization, we first solve the nonlinear system proposed by Pollock [RAM, Eq. (20)]:

$$f(x) = \begin{bmatrix} (x_1 - 1) + (x_2 - 3)^2 \\ \varepsilon(x_2 - 3) + \frac{3}{2}(x_1 - 1)(x_2 - 3) + (x_2 - 3)^2 + (x_2 - 3)^3 \end{bmatrix}, \tag{2.2e}$$

with Jacobian

$$J_f(x) = \begin{bmatrix} 1 & 2(x_2 - 3) \\ \frac{3}{2}(x_2 - 3) & \varepsilon + \frac{3}{2}(x_1 - 1) + 2(x_2 - 3) + 3(x_2 - 3)^2 \end{bmatrix}. \tag{2.2f}$$

This benchmark validates Newton's method under small-parameter perturbations and provides a controlled setting before moving on to PDE-based problems.
Phase-specific forms:

$$-ku_x(1) = \alpha(u - u_\star) + g \qquad \text{(Phase 0)} \tag{2.2g}$$

$$-ku_x(1) = \alpha(u^4 - u_\star^4) + g \qquad \text{(Phases 1–2)} \tag{2.2h}$$

$$-ku_x(1,t) = \alpha\big(u^4(1, t - \Delta t) - u_\star^4\big) + g \qquad \text{(Phase 3)} \tag{2.2i}$$

$$-ku_x(1,t) = \alpha(u^4(1,t) - u_\star^4) + g \qquad \text{(Phase 4)} \tag{2.2j}$$

2.3 **Spatial Discretization.** The domain is discretized with spacing $h = 1/N$, nodes $x_j = jh$. Interior stencil:

$$u_{xx}(x_j) \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2}. \tag{2.3a}$$

Boundary derivative approximations:

$$u_x(1) \approx \tfrac{u_N - u_{N-1}}{h} \qquad \text{(FD)} \tag{2.3b}$$

$$u_x(1) \approx \tfrac{u_{N+1} - u_{N-1}}{2h}, \qquad \text{(CCFD, ghost point)} \tag{2.3c}$$

with u_{N+1} eliminated using the radiation condition.
2.4 **Time Discretization.** For Phases 3–4 we apply Backward Euler:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{k}{c} \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2}. \tag{2.4a}$$

Boundary handling:

$$-ku_x(1, t^{n+1}) \approx \alpha\big((u_N^n)^4 - u_\star^4\big) + g \qquad \text{(Phase 3)} \tag{2.4b}$$

$$-ku_x(1, t^{n+1}) \approx \alpha\big((u_N^{n+1})^4 - u_\star^4\big) + g \qquad \text{(Phase 4)} \tag{2.4c}$$

2.5 **Nonlinear Solver.** Nonlinear boundary terms are handled with Newton's method. Generic update:

$$U^{(k+1)} = U^{(k)} - J^{-1}(U^{(k)})F(U^{(k)}), \tag{2.5a}$$

where $F(U)$ is the residual and $J(U)$ its Jacobian. Residual contributions at the boundary use

$$N(U_N) = \frac{\alpha}{h}(U_N^4 - u_\star^4) + \frac{g}{h}, \qquad \frac{dN}{dU_N} = \frac{4\alpha}{h}U_N^3.$$

2.6 **Phase Summaries.**

- **Phase 0:** Linear steady-state, Robin BC (2.2g). FD (first-order) and CCFD (second-order) schemes compared against exact solution $u(x) = e^{2x}$.
- **Phase 1:** Pollock benchmark system (2.2e)–(2.2f) (no PDE discretization), used to validate Newton's method on a nonlinear algebraic problem.
- **Phase 2:** Steady nonlinear boundary problem (2.2h), solved with Newton + analytic Jacobian.
- **Phase 3:** Time-dependent with lagged nonlinear BC (2.2i) together with Backward Euler (**??**), produces linear solves each step.
- **Phase 4:** Fully implicit nonlinear PDE (2.2j) with Backward Euler (2.4c), solved each timestep with Newton; stable even for large α.

2.7 **Implementation Notes.** MATLAB code is modular by phase. Residual/Jacobian functions encapsulate discretization and boundary terms; ghost-point elimination is built into the CCFD functions. Newton iteration includes damping and residual checks. Time-stepping scripts call Newton at each step (Phase 4). Console logs report residual norms and iteration counts, and plots verify convergence.

3. Numerical Methods

3.1 **Spatial Discretization.** We discretize the spatial domain using finite differences on a uniform grid of size h. The schemes applied here are defined in Section 2.3.

- **Interior Points:** All phases use a second-order centered finite difference for u_{xx}, as shown in Equation (2.3a).
- **FD Method (First-Order):** A one-sided backward difference is used for u_x at the right boundary, as in Equation (2.3b). This method is simple to implement but only first-order accurate near the boundary.
- **CCFD Method (Second-Order):** A ghost point u_{N+1} is introduced and eliminated using the boundary condition, yielding the scheme shown in Equation (2.3c). This approach provides second-order accuracy at the boundary.

3.2 **Time Discretization.** For time-dependent phases (Phases 3–4), we use the Backward Euler method with timestep Δt, which is unconditionally stable and first-order accurate.

- **Interior Updates:** At each interior point, the update rule matches Equation (2.4a), which approximates the time derivative using backward Euler and the spatial second derivative using centered finite differences.
- **Phase 3 (Lagged BC):** The nonlinear boundary term is evaluated explicitly using u^n. This results in a linear system at each time step that can be solved directly.
- **Phase 4 (Fully Implicit):** The nonlinear boundary term is evaluated at t^{n+1}, leading to a nonlinear system that is solved using Newton's method with analytic Jacobian assembly.

3.3 **Phase Summary Table.**

Phase	Description	Discretization	Solver
0	Linear Robin (steady-state)	FD, CCFD	Direct
1	Newton benchmark (Pollock)	N/A	Newton
2	Nonlinear Robin (steady-state)	FD, CCFD	Newton
3	Time-lagged nonlinear BC	FD, CCFD + Backward Euler	Direct
4	Fully implicit nonlinear PDE	FD, CCFD + Backward Euler	Newton

4. Nonlinear Solver Implementation

4.1 **Newton's Method.** To solve the nonlinear systems arising in Phases 1, 2, and 4, we implement Newton's method. At each iteration k, the solution vector is updated using the classical Newton step (see Equation (2.5a)), where $F(U)$ is the residual and $J(U)$ is the Jacobian matrix. For all PDE-based phases, the Jacobian is constructed analytically to improve accuracy and computational efficiency.

Newton's method is applied in the following contexts:

- **Phase 1:** Scalar Newton iteration is validated on a nonlinear benchmark system (Pollock's Equation (20)).
- **Phase 2:** Newton's method solves the steady-state PDE with a nonlinear radiation boundary.
- **Phase 4:** Newton's method is used at each time step to solve the fully implicit nonlinear diffusion system.

To ensure robustness, the Newton solver includes adaptive damping and residual norm checks. The solver logs residual norms and iteration counts to the console to monitor convergence.

4.2 **Modular Boundary Flux Function.** The radiation boundary condition introduces a nonlinear flux term at the right boundary of the domain. For use in both the residual and Jacobian, we define a modular function (see Section 2.5):

$$N(U_N) = \frac{\alpha}{h}(U_N^4 - u_\star^4) + \frac{g}{h}, \qquad \frac{dN}{dU_N} = \frac{4\alpha}{h}U_N^3.$$

This modular form is used in both the FD and CCFD boundary treatments for:

- **Phase 2:** Steady-state nonlinear problem, where U_N denotes the boundary temperature at steady state.
- **Phase 4:** Fully implicit time-dependent problem, where U_N represents the temperature at the right boundary at time t^{n+1}.

In each case, the function $N(U_N)$ contributes directly to the final row of the residual vector, and its derivative is inserted into the corresponding Jacobian entry. This modular design improves maintainability and ensures consistency across solver implementations.

5. MATLAB Code Structure

5.1 **Organization.** Each phase resides in its own folder (see Section 1.3) with a driver script and modular functions for residuals, Jacobians, and boundary fluxes. The structure supports:

- Switching between FD (first-order) and CCFD (second-order) discretizations
- Reusable boundary condition routines with consistent interfaces
- A shared Newton solver (`newton_eq20.m`) across nonlinear phases

- Parameter control from the top of each driver script ($\alpha, u_\star, g, N, \Delta t$, etc.)

Driver scripts print iteration progress and generate plots automatically.

Phase	`Run Command(s)`	Description / Outputs
0	`>> robin`	FD vs. CCFD convergence on a manufactured solution; log–log error plots over grid sizes.

1	`>> convergence_plot`
	`>> compare_convergence_epsilons`
	`>> animate_convergence_epsilons`
		Pollock benchmark for Newton's method. Tracks convergence vs. iteration, compares error decay across ε, and animates convergence domains. Validates Newton's stability under perturbations.

2	`>> robin_phase2_convergence`	Nonlinear steady-state PDE with radiation boundary. FD/CCFD solvers using `f_and_J_robin_fd.m` and `f_and_J_robin_ccfd.m`; solves for $\alpha \in \{0.1, 1.0, 10.0\}$, grid sizes $h \in \{0.1, 0.05, 0.025, 0.0125\}$. Computes ℓ^∞ errors vs. reference and produces log–log convergence plots.
3	`>> robin_phase3`	Time-dependent PDE with time-lagged nonlinear radiation boundary. Solves with FD and CCFD; boundary enforced using previous timestep. Outputs solution plots and FD–CCFD comparison at final time.
4	`>> phase4_compare`	Fully implicit nonlinear PDE with radiation boundary. Uses `f_and_J_phase4_fd.m`, `f_and_J_phase4_ccfd.m`, and `newton_eq20.m` each timestep. Produces final solutions and prints $\|\|\text{FD} - \text{CCFD}\|\|_\infty$.

5.2 **Parameter Tuning.**

- Adjustable at the top of each driver: grid resolution N, time step Δt, radiation coefficient α, reference temperature $u_\star$.
- Newton damping options inside `newton_eq20.m`.
- Grid refinement studies supported in Phases 2 and 4 via looped runs over N.
- Default parameters reproduce validated results; users may vary them to explore solver behavior.

6. Results and Observations

6.1 **Phase 0 (Convergence Study).** To verify the order of accuracy for the two spatial boundary treatments, we solve a linear Poisson problem with Robin boundary conditions using the exact solution

$$u(x) = e^{2x}, \quad -u''(x) = f(x), \quad -u'(0) = \alpha_0 u(0) + g_0, \quad -u'(1) = \alpha_1 u(1) + g_1.$$

The forcing function $f(x)$ and boundary parameters are chosen so that this exact solution holds. We test:

- D_-: backward difference for flux, expected first-order accuracy.
- D_0: centered difference with ghost point, expected second-order accuracy.

Error Table: Errors in both norms are reported across four grid spacings.

h	$\|e\|_\infty^{D_-}$	$\|e\|_2^{D_-}$	$\|e\|_\infty^{D_0}$	$\|e\|_2^{D_0}$
10^{-1}	5.56e-01	4.43e-01	2.29e-02	1.14e-02
10^{-2}	5.77e-02	4.31e-02	2.30e-04	1.02e-04
10^{-3}	5.79e-03	4.30e-03	2.30e-06	1.01e-06
10^{-4}	5.79e-04	4.29e-04	2.27e-08	9.86e-09

Convergence Rates:

h	$\text{Rate}_\infty^{D_-}$	$\text{Rate}_2^{D_-}$	$\text{Rate}_\infty^{D_0}$	$\text{Rate}_2^{D_0}$
10^{-1}	0.98	1.01	2.00	2.05
10^{-2}	1.00	1.00	2.00	2.01
10^{-3}	1.00	1.00	2.00	2.01

Log–log Error Plots: Figure A.1 confirms the expected first- and second-order rates in both norms.

6.2 **Phase 1 (Pollock Benchmark).** We study Newton's method on Pollock's benchmark nonlinear system:

$$f(x_1, x_2) = \begin{bmatrix} (x_1 - 1) + (x_2 - 3)^2 \\ \varepsilon(x_2 - 3) + \frac{3}{2}(x_1 - 1)(x_2 - 3) + (x_2 - 3)^2 + (x_2 - 3)^3 \end{bmatrix}.$$

This system has a stable root at $(1, 3)$ and an unstable root that varies with ε.
Root Verification: For $\varepsilon = 10^{-6}$, our Newton solver with analytic Jacobian and damping converges to the stable root $(1, 3)$ in a small number of iterations.
Convergence Domain Analysis: Grid sweeps of initial guesses show how the basin of attraction contracts as $\varepsilon \to 0$. As ε decreases, more iterations are required and convergence sensitivity increases.
Role in Project: This benchmark validated that the Newton infrastructure (Jacobian assembly + damping) works under nonlinear conditions. These routines were reused directly in Phases 2–4.

6.3 **Phase 2 (Steady Nonlinear Boundary Condition).** We solve

$$-ku''(x) = f(x), \quad x \in (0, 1), \quad u(0) = 0, \quad -ku'(1) = \alpha(u(1)^4 - u_\star^4) + g,$$

with $f(x) \equiv 0$, $g = 0$, and $u_\star = 1$. The interior solution is linear, $u(x) = Ax$, with slope A set by the nonlinear boundary.
Discretizations:

- **FD:** backward difference at the right boundary.
- **CCFD:** centered difference with ghost point.

Iteration Counts: Both schemes converged in 4–5 Newton iterations for each α.

α	FD Iterations	CCFD Iterations
0.1	4	4
1.0	5	5
10.0	4	4

Grid Refinement Study:

- FD shows first-order accuracy ($\mathcal{O}(h)$).
- CCFD, although based on a second-order stencil, also shows global $\mathcal{O}(h)$ accuracy due to boundary error dominating.

Refinement Plots: Figure A.2 confirms first-order convergence in both schemes.
Takeaways:

- Nonlinear Robin condition enforced accurately by both methods.
- Newton's method converged reliably even for large α.
- Global error governed by boundary, producing first-order convergence.

6.4 **Phase 3 (Time-Lagged Nonlinear Radiation Boundary).** We extend Phase 2 to the time-dependent PDE

$$c\,u_t = k\,u_{xx}, \quad x \in (0,1), \quad t > 0,$$

with explicit (time-lagged) nonlinear Robin condition:

$$-k\,u_x(1,t) = \alpha\left(u^4(1,t-\Delta t) - u_\star^4\right) + g.$$

Schemes Tested:

- **FD:** backward difference at the boundary.
- **CCFD:** ghost-point method.

Simulation Setup:

1. Short-time case: $\Delta t = 0.001, T = 0.5$.
2. Long-time case: $\Delta t = 0.0001, T = 3.0$.

Error Quantification: Sup-norm differences at final time:

α	Short-Time ($T = 0.5$)	Long-Time ($T = 3.0$)
0.1	8.618e-03	5.609e-05
1.0	6.180e-04	1.343e-08
10.0	1.787e-01	3.109e-15

Observations:

- Both FD and CCFD remain stable for all parameters.
- CCFD profiles are smoother; FD appears slightly more diffusive.

- Both converge to the same steady-state limit across all α.

Figures: Solution profiles for short- and long-time cases appear in Figures A.3–A.4.

6.5 **Phase 4 (Fully Implicit Nonlinear Radiation Boundary).** We remove the time-lag and enforce the nonlinear radiation condition implicitly at each step:

$$c\,u_t = k\,u_{xx}, \quad x \in (0,1), \quad t > 0,$$

$$-k\,u_x(1,t) = \alpha(u^4(1,t) - u_\star^4) + g, \quad u(0,t) = 0.$$

Residual Formulations:

- **FD:**
$$F_i^{\text{FD}}(U) = \begin{cases} \frac{c}{\Delta t}(U_i - U_i^{\text{prev}}) - kD^2U_i, & 2 \le i \le N-1, \\ \frac{c}{\Delta t}(U_N - U_N^{\text{prev}}) - k\frac{U_N - U_{N-1}}{h^2} + \frac{1}{h}N(U_N), & i = N, \end{cases}$$
where $N(U_N) = \alpha(U_N^4 - u_\star^4) + g$.

- **CCFD:**
$$F_N^{\text{CCFD}}(U) = \tfrac{c}{\Delta t}(U_N - U_N^{\text{prev}}) - k\tfrac{U_{N+1} - 2U_N + U_{N-1}}{h^2},$$
with ghost point eliminated via
$$\tfrac{U_{N+1} - U_{N-1}}{2h} = -\tfrac{1}{k}\big[\alpha(U_N^4 - u_\star^4) + g\big].$$

Simulation Parameters: $c = k = u_\star = 1$, $g = 0$, initial condition $u(x,0) = \sin(\pi x)$, final time $T = 0.1$.
Results:

- Newton converged to tolerance 10^{-8} at every time step.
- Both FD and CCFD produced stable, smooth evolution to steady state.
- Large α and large Δt remained stable, unlike explicit Phase 3.
- Final steady-state profiles matched Phase 2 solutions.

Figures: Solution comparisons for $\alpha = 1.0$ and $\alpha = 10.0$ appear in Figure A.5.

7. Conclusion

This project developed a complete numerical solver pipeline for one-dimensional heat equations with nonlinear radiation boundary conditions. Beginning with a steady-state linear model in Phase 0, each successive phase introduced additional mathematical and computational complexity, culminating in a fully implicit time-dependent solver in Phase 4.

Across all phases, finite difference methods were used for spatial discretization, Newton's method for resolving nonlinearities, and a modular code structure for assembling residuals and Jacobians. These strategies ensured stability, convergence, and clarity of implementation throughout.

Phase 4 served as the capstone of the project, integrating all previous components into a robust solver for nonlinear parabolic problems. Both FD and CCFD boundary schemes were implemented and tested, with results confirming agreement between methods and stable convergence even for stiff boundary conditions. Numerical experiments further validated that the time-dependent solution converges to the Phase 2 steady

state, confirming both physical fidelity and numerical correctness.

The final implementation is modular, validated, and consistent with the theoretical behavior of nonlinear heat transfer problems with radiation effects.

APPENDIX A (SUPPLEMENTARY FIGURES)

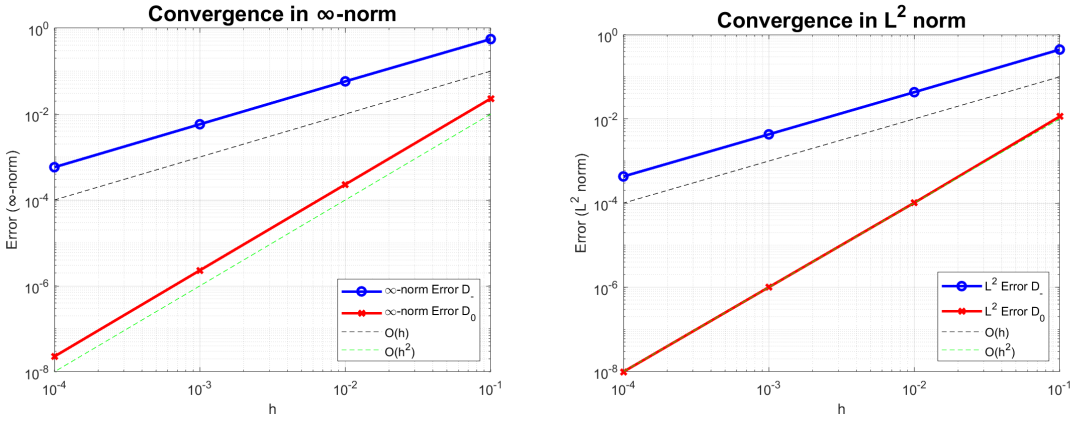

Figure A.1: Phase 0: Log–log convergence plots for D_- and D_0 in the ∞-norm (left) and L^2 norm (right).

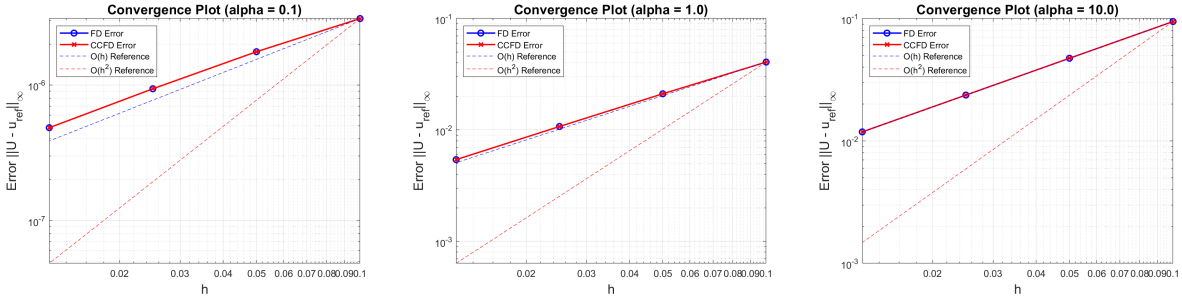

Figure A.2: Phase 2: Grid refinement plots for $\alpha = 0.1$, 1.0, and 10.0. First-order convergence is observed.

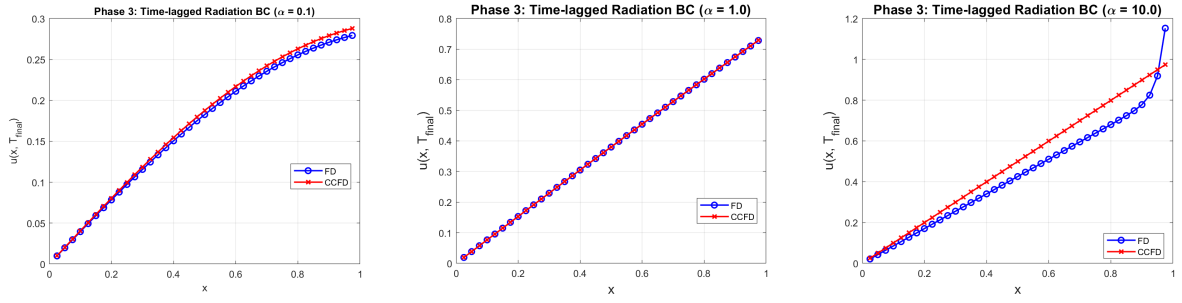

Figure A.3: Phase 3: Short-Time Simulation ($T_{\text{final}} = 0.5$, $\Delta t = 0.001$) for various α.

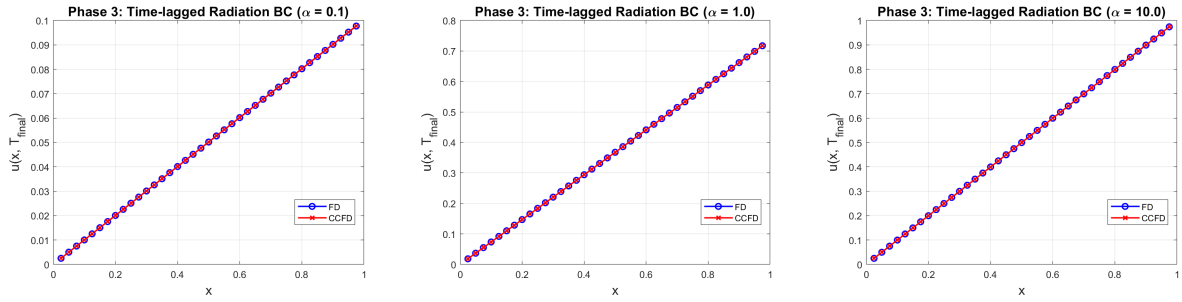

Figure A.4: Phase 3: Long-Time Simulation ($T_{\text{final}} = 3.0$, $\Delta t = 0.0001$) for various α.

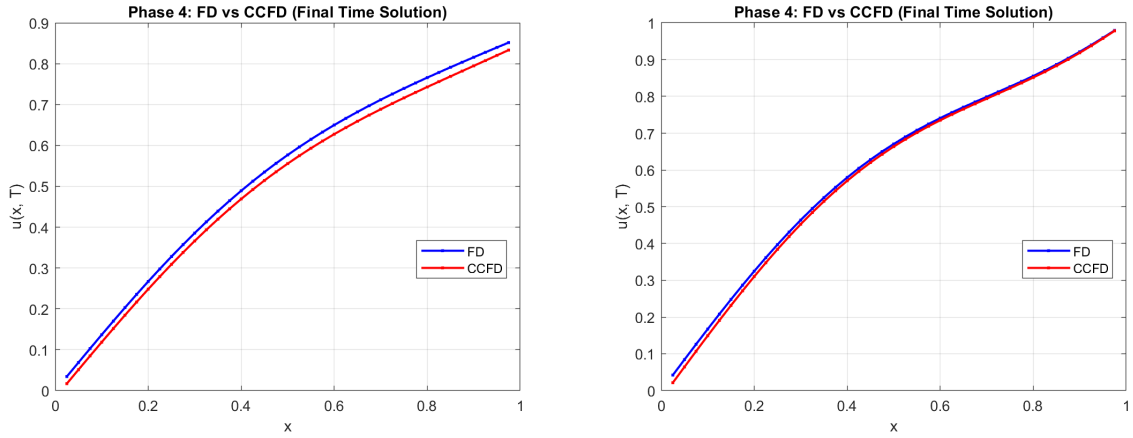

Figure A.5: Phase 4: FD vs. CCFD final solutions for (left) $\alpha = 1.0$, $\Delta t = 10^{-4}$ and (right) $\alpha = 10.0$, $\Delta t = 0.05$.

APPENDIX B (SELECTED MATLAB CODE)

This appendix presents two representative MATLAB code segments used in Phase 4 of the project. The first implements an adaptive damped Newton solver that was reused across all nonlinear problems. The second is the Phase 4 driver script that evolves both FD and CCFD solutions and compares their final profiles. Together, they demonstrate the modular solver design and boundary treatment logic central to the project.

Listing 1: Adaptive damped Newton's method

```
function [x, num_iter, converged] = newton_eq20(x0, F_and_J, tol,
max_iter)
    x = x0;
    converged = false;
    [F, ~] = F_and_J(x);
    res_old = norm(F, 2);
    for k = 1:max_iter
        [F, JF] = F_and_J(x);
        res = norm(F, 2);
        if res < tol
```

```
            converged = true;
            break;
        end
        fprintf('Iter %d: ||F|| = %.2e\n', k, res);
        dx = JF \ F;
        beta_k = 1;
        x_new = x - beta_k * dx;
        [F_new, ~] = F_and_J(x_new);
        res_new = norm(F_new, 2);
        while res_new > res && beta_k > 1e-4
            beta_k = beta_k / 2;
            x_new = x - beta_k * dx;
            [F_new, ~] = F_and_J(x_new);
            res_new = norm(F_new, 2);
        end
        if res_new >= res
            warning('Damping failed to reduce residual. Stopping.');
            break;
        end
        x = x_new;
        res_old = res_new;
    end
    num_iter = k;
end
```

Listing 2: Phase 4: Comparison of FD and CCFD solvers

```
function phase4_compare()
    k = 1; c = 1; alpha = 10; g = 0; u_star = 1;
    h = 1/40; N = round(1/h) - 1;
    x_full = linspace(0, 1, N + 2)';
    x = x_full(2:end-1);
    dt = 0.05; T_final = 0.1; Nt = round(T_final / dt);
    tol = 1e-8; max_iter = 50;
    U_fd   = sin(pi * x);
    U_ccfd = sin(pi * x);
    for n = 1:Nt
        U_prev = U_fd;
        FJ = @(U_new) f_and_J_phase4_fd(U_new, U_prev, alpha, g,
        u_star, h, k, c, dt);
        [U_fd, ~, conv] = newton_eq20(U_prev, FJ, tol, max_iter);
    end
    for n = 1:Nt
        U_prev = U_ccfd;
        FJ = @(U_new) f_and_J_phase4_ccfd(U_new, U_prev, alpha, g,
        u_star, h, k, c, dt);
        [U_ccfd, ~, conv] = newton_eq20(U_prev, FJ, tol, max_iter);
    end
end
```

REFERENCES

[1] Sara Pollock and Hunter Schwartz. "Benchmarking results for the Newton–Anderson method," *Results in Applied Mathematics*, 8:100095, 2020.

[2] Malgorzata Peszynska. "Modeling nonlinear radiation boundary conditions in heat conduction problems," Lecture notes and project instructions, Oregon State University, 2025.

[3] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems*. SIAM, 2007.

[4] C. T. Kelley. *Solving nonlinear equations with Newton's method*. SIAM, 2003.

[5] J. David Logan. *Applied mathematics*, 3rd edition. Wiley, 2006.

[6] MATLAB Documentation. MathWorks Inc., 2025. Available at: `https://www.mathworks.com/help/`

[7] Notes from MTH 452 and MTH 453 (Numerical ODEs and PDEs). Oregon State University, 2024–2025.