A Coupled 1D Mixed-Form Richards and Catchment-Scale Hydrologic Model for Simulating Vadose Zone Dynamics in Engineered Drainage Systems

Maria Castro, Marcus Nóbrega Gomes Jr.

May 16, 2025

Contents

Author Information

Marcus N. Gomes Jr.
Ph.D. - Postdoctoral Researcher II at University of Arizona

Website:	marcusnobrega-eng.github.io/profile
C.V.:	CV
ORCID:	https://orcid.org/0000-0002-8250-8195
GitHub:	https://github.com/marcusnobrega-eng
Email:	marcusnobrega.engcivil@gmail.com

1 Introduction

Hydrological modeling of the vadose zone — the unsaturated region between the land surface and the water table — is essential for understanding water and solute movement in both natural and engineered systems. In urban environments, where impermeable surfaces, compacted soils, and anthropogenic structures complicate infiltration dynamics, physically based models offer valuable tools for design, evaluation, and long-term planning of stormwater management infrastructure.

This manual presents a modular and extensible MATLAB implementation of a physically based hydrologic model that solves the mixed-form Richards equation in one dimension (1D), fully coupled with a catchment-scale runoff generation module. The model is designed for simulating vertical water flow through variably saturated porous media, accounting for multilayer soil structures, evaporation, surface fluxes, and engineered drainage features such as orifices and spillways.

1.1 Motivation and Applications

While many infiltration models rely on empirical relationships (e.g., Horton, Green-Ampt), these are often inadequate in heterogeneous soils or where strong capillary effects, layered materials, or structural drainage features are present. A physically based model like the mixed-form Richards equation captures the nonlinearity and state-dependence of hydraulic conductivity and water retention, allowing simulations to remain valid across dry and wet ranges.

This model is particularly suited for applications in:

- **Sustainable Drainage Systems (SuDS)**, including bioretention cells, infiltration basins, and bioswales
- **Permeable pavement systems**, composed of gravel, sand, and asphalt layers
- **Urban catchments** with surface runoff, impervious routing, and structural overflows

- **Experimental validation** against column experiments or analytical benchmarks (e.g., Celia et al., 1990)

1.2 Model Structure Overview

The modeling system is modular and organized into the following components:

1. **Richards Equation Solver** – A 1D vertical solver for unsaturated flow, using the mixed-form formulation with adaptive time stepping and Newton-Raphson iterations.
2. **Catchment Hydrology Module** – Computes time-varying surface fluxes using rainfall, curve number abstraction (SCS-CN), impervious runoff, evapotranspiration, and structural drainage.
3. **Drainage Structures** – Simulates flow through embedded orifices and spillways, with optional capping to enforce volume-based constraints.
4. **Post-Processing Suite** – Generates publication-quality plots, animations, and exports including pressure head, water content, Darcy fluxes, flow duration curves, and wetting front tracking.
5. **Input–Output Automation** – Reads inputs from '.mat' or Excel files, logs simulation metadata, and exports results to organized folders for diagnostics and analysis.

1.3 Key Features

- Fully supports **layered media**, including coarse layers (e.g., gravel) and fine-textured soils.
- Supports both **flux- and head-based boundary conditions**, including dynamic ponding logic.
- Includes optional modules for **pollutant transport** using empirical buildup/washoff models.
- Designed for integration into **urban hydrologic planning**, particularly green infrastructure design.
- Mass balance is rigorously tracked at each time step with **diagnostic error reporting**.

1.4 Manual Organization

This manual serves both as a scientific reference and a user guide. It is organized into the following sections:

1. **Governing Equations** – Physics of the mixed-form Richards equation and soil hydraulic functions.
2. **Numerical Methods** – Discretization schemes, solver strategies, and Jacobian handling.
3. **Model Architecture** – Folder structure and function-level overview.
4. **Input Configuration** – Description of input files, parameters, and scenario setup.
5. **Execution Guide** – How to run the model, log simulations, and track ponding and drainage.
6. **Output and Diagnostics** – Exported variables, figures, animations, and error metrics.
7. **Validation Benchmarks** – Recommended test cases and validation references.
8. **Case Studies** – Example applications to permeable pavements, gravel columns, and SuDS.
9. **Troubleshooting** – Solver stability, mass balance failures, and BC misconfigurations.
10. **Appendices** – Van Genuchten parameter tables, dimensional analysis, and full function index.

The modeling framework consists of modular MATLAB functions (see Table 1) organized by numerical, physical, and I/O functionality.

Table 1: Core model functions and their purpose.

Function Name	Description	Folder
`Main_Solver.m`	Main numerical engine for solving the mixed-form Richards equation via Newton-Raphson with adaptive time stepping	`Main_Function`
`Mixed_Richards_Model_1D.m`	Top-level launcher script: loads input, calls solver, post-processes results	`Main_Function`
`Input_Data.m`	Initializes simulation name, mesh, boundary conditions, soil properties, and output folders	`Model_Configurations`
`compute_residual.m`	Computes residual vector and Darcy fluxes for Newton iterations	`Numerical_Solver`
`compute_jacobian.m`	Assembles Jacobian matrix via finite differences (used in Newton solver)	`Numerical_Solver`
`drainage_sinks.m`	Computes orifice and spillway flows, including capped drainage and analytical derivatives	`Physical_Functions_And_Utilities`
`get_boundary_values.m`	Calculates boundary conditions dynamically at each time step (handles ponding, evaporation, rainfall)	`Physical_Functions_And_Utilities`
`theta_vgm.m`	Computes water content $\theta(h)$ using Van Genuchten equation	`Physical_Functions_And_Utilities`

(continued on next page)

Function Name	Description	Folder
K_vgm.m	Computes $K(h)$ using Van Genuchten–Mualem formulation	Physical_Functions_And_Utilities
mass_balance_check.m	Evaluates mass conservation errors and cumulative fluxes at each time step	Physical_Functions_And_Utilities
generate_nonlinear_mesh.m	Generates refined 1D mesh toward surface (Hydrus-style stretching)	Physical_Functions_And_Utilities
plot_soil_profiles.m	Plots vertical profiles of head, moisture, and flux at given time steps	Physical_Functions_And_Utilities
plot_vg_retention_curves.m	Plots analytical $\theta(h)$, $K(h)$, and $K(\theta)$ curves from soil parameters	Physical_Functions_And_Utilities
feddes_alpha.m	Computes evaporation reduction coefficient based on surface suction (Feddes function)	Physical_Functions_And_Utilities
Rainfall_ETP_Timeseries.m	Prepares rainfall and evapotranspiration data, including event extraction and aridity index	Forcing
Catchment_Outputs.m	Runs SCS-CN runoff or loads hydrograph to compute Neumann flux boundary condition	Forcing
SCS_Hydrologic_Model.m	Implements modified SCS Curve Number runoff generation model	Forcing
Buildup_Washoff_Model.m	Computes pollutant buildup and washoff from storm events and inflow hydrograph	Forcing
SaveSimulationResults.m	Saves outputs to Excel (.xlsx) with time axes and structured tables	Main_Function
Post_Processing.m	Master script to generate time series, profiles, FDCs, wetting front plots	Main_Function
generate_richards_animations.m	Exports MP4 animations of $\theta(z,t)$, $h(z,t)$, and $q(z,t)$ over time	Visualization_And_Output
analyze_richards_outputs.m	Alternative visualization of profiles and time series for diagnostics	Visualization_And_Output
plot_flow_duration_curves.m	Plots upward and downward flow duration curves (log scale, by depth)	Visualization_And_Output

2 Numerical Methods

2.1 Overview

The governing equation for variably saturated vertical flow in the soil profile is the **mixed-form Richards equation**:

$$\frac{\partial \theta(h)}{\partial t} + S_s \frac{\partial h}{\partial t} = \frac{\partial}{\partial z}\left[K(h)\left(\frac{\partial h}{\partial z} + 1\right)\right] + S(z,t) \tag{1}$$

where:

- h is pressure head [m]

- $\theta(h)$ is volumetric water content [m^3/m^3]
- $K(h)$ is unsaturated hydraulic conductivity [m/s]
- S_s is specific storage [1/m]
- z is vertical coordinate, positive upward [m]
- $S(z,t)$ is source/sink term [1/s]

This equation is solved using a fully implicit backward Euler scheme combined with Newton–Raphson iteration, line search, and adaptive time stepping.

2.2 Soil Water Retention and Conductivity Functions

The model uses the Van Genuchten–Mualem constitutive relationships to relate pressure head h to both volumetric water content $\theta(h)$ and hydraulic conductivity $K(h)$. These empirical functions are widely used in vadose zone hydrology for their smooth behavior and good agreement with experimental data.

Volumetric Water Content $\theta(h)$

The soil moisture retention curve is given by:

$$\theta(h) = \begin{cases} \theta_r + (\theta_s - \theta_r)\left[1 + (\alpha|h|)^n\right]^{-m}, & h < 0 \\ \theta_s, & h \geq 0 \end{cases} \tag{2}$$

- θ_r – residual water content [–]
- θ_s – saturated water content [–]
- α – inverse air-entry pressure [1/m]
- n – pore-size distribution parameter [–]
- $m = 1 - \frac{1}{n}$ – shape factor

This function is continuous and differentiable for $h < 0$, and guarantees $\theta(h) \to \theta_s$ as $h \to 0^-$.

Effective Saturation

The effective saturation S_e is defined as:

$$S_e(h) = \frac{\theta(h) - \theta_r}{\theta_s - \theta_r} \tag{3}$$

This normalizes the moisture curve for use in conductivity and other state-dependent functions.

Hydraulic Conductivity $K(h)$

The unsaturated hydraulic conductivity is computed via the Mualem model:

$$K(h) = K_s \cdot S_e^{1/2} \left[1 - \left(1 - S_e^{1/m}\right)^m\right]^2 \tag{4}$$

- K_s – saturated hydraulic conductivity [m/s]
- S_e – effective saturation [–]

This equation ensures that $K(h) \to K_s$ as $h \to 0^-$ and $K(h) \to 0$ as $h \to -\infty$.

Numerical Considerations

- The model ensures bounded values of S_e and enforces $\theta_r \leq \theta(h) \leq \theta_s$
- Derivatives of $\theta(h)$ and $K(h)$ with respect to h are used in Jacobian estimation
- For $h \geq 0$, $\theta(h) = \theta_s$ and $K(h) = K_s$

These constitutive relationships are implemented in:

- `theta_vgm.m` – computes $\theta(h)$
- `K_vgm.m` – computes $K(h)$

2.3 Spatial Discretization

We divide the vertical profile into N_z control volumes with thickness Δz_i [m], indexed from bottom ($i = 1$) to top ($i = N_z$). The spatial fluxes are computed using finite volume approximation:

$$q_{i+1/2} = -K_{i+1/2}\left(\frac{h_{i+1} - h_i}{\Delta z_{i+1/2}} + 1\right) \tag{5}$$

with:

$$K_{i+1/2} = \frac{K(h_i) + K(h_{i+1})}{2} \quad [\text{m/s}]$$

$$\Delta z_{i+1/2} = \frac{\Delta z_i + \Delta z_{i+1}}{2} \quad [\text{m}]$$

Flux divergence is approximated as:

$$\left.\frac{\partial q}{\partial z}\right|_i \approx \frac{q_{i+1/2} - q_{i-1/2}}{\Delta z_i} \quad [\text{s}^{-1}] \tag{6}$$

2.4 Temporal Discretization

We use the fully implicit (backward Euler) scheme:

$$\frac{\partial \theta}{\partial t} \approx \frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} \quad [1/\text{s}] \tag{7}$$

$$\tag{8}$$

$$\frac{\partial h}{\partial t} \approx \frac{h_i^{n+1} - h_i^n}{\Delta t} \quad [\text{m/s}] \tag{9}$$

Substituting into Eq. (1), the residual for node i becomes:

$$F_i = \frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} + S_s \cdot \frac{h_i^{n+1} - h_i^n}{\Delta t} + \frac{q_{i+1/2}^{n+1} - q_{i-1/2}^{n+1}}{\Delta z_i} - S_i^{n+1} = 0 \tag{10}$$

The nonlinear residual vector $\mathbf{F}(\mathbf{h}) = \{F_1, ..., F_{N_z}\}$ is the target of the solver.

2.5 Newton–Raphson Iteration with Line Search

At each time step, we solve the nonlinear system $\mathbf{F}(\mathbf{h}) = \mathbf{0}$ using Newton–Raphson iterations:

$$\mathbf{J}^k \cdot \Delta \mathbf{h}^k = -\mathbf{F}^k \tag{11}$$

$$\mathbf{h}^{k+1} = \mathbf{h}^k + \lambda \cdot \Delta \mathbf{h}^k \tag{12}$$

Where:

- $\mathbf{J}^k$ is the Jacobian matrix at iteration k [1/m]
- $\Delta\mathbf{h}^k$ is the Newton update direction [m]
- λ is a line search step size $0 < \lambda \leq 1$

To enhance convergence robustness, we implement the **Armijo line search**:

- Start with $\lambda = 1$
- Reduce $\lambda \to \beta\lambda$ (e.g., $\beta = 0.5$) until:

$$\|\mathbf{F}(\mathbf{h} + \lambda\Delta\mathbf{h})\|_2 \leq (1 - \alpha\lambda)\|\mathbf{F}(\mathbf{h})\|_2$$

- Typical $\alpha = 10^{-4}$

This ensures sufficient decrease in the residual norm and guards against overshooting.

2.6 Jacobian Approximation

The Jacobian matrix $\mathbf{J}$ is computed numerically using forward finite differences:

$$J_{:,j} \approx \frac{\mathbf{F}(\mathbf{h} + \varepsilon \cdot \mathbf{e}_j) - \mathbf{F}(\mathbf{h})}{\varepsilon} \tag{13}$$

Where:

- $\varepsilon \approx 10^{-6}$ [m]
- $\mathbf{e}_j$ is the unit vector at position j

Because F_i depends only on h_{i-1}, h_i, and h_{i+1}, the Jacobian is sparse and tridiagonal in structure (except near boundaries).

2.7 Symbolic Jacobian Structure for a 5-Node System

We now present a symbolic derivation of the Jacobian matrix $\mathbf{J} = \partial\mathbf{F}/\partial\mathbf{h}$ for a discretized Richards model with $N_z = 5$ nodes. The system residual vector is:

$$\mathbf{F}(\mathbf{h}) = \begin{bmatrix} F_1(h_1, h_2) \\ F_2(h_1, h_2, h_3) \\ F_3(h_2, h_3, h_4) \\ F_4(h_3, h_4, h_5) \\ F_5(h_4, h_5) \end{bmatrix}$$

Each residual F_i depends on three adjacent heads: h_{i-1}, h_i, h_{i+1}, except at boundaries.

—

Case A: Dirichlet–Dirichlet

Top and bottom pressure heads are fixed. Residuals at boundaries enforce those values directly:

$$F_1 = h_1 - h_{\text{bot}}, \quad F_5 = h_5 - h_{\text{top}}$$

The Jacobian matrix is:

$$\mathbf{J}_{\text{Dir–Dir}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Where: - $a_i = \partial F_i / \partial h_{i-1}$ - $b_i = \partial F_i / \partial h_i$ - $c_i = \partial F_i / \partial h_{i+1}$

—

Case B: Neumann–Neumann

Top and bottom fluxes are specified. Residuals at boundaries are flux balances, not algebraic constraints:

$$F_1 = \frac{\theta_1^{n+1} - \theta_1^n}{\Delta t} + S_{s,1} \frac{h_1^{n+1} - h_1^n}{\Delta t} + \frac{q_{3/2} - q_{\text{bot}}}{\Delta z_1} - S_1$$

$$F_5 = \frac{\theta_5^{n+1} - \theta_5^n}{\Delta t} + S_{s,5} \frac{h_5^{n+1} - h_5^n}{\Delta t} + \frac{q_{\text{top}} - q_{9/2}}{\Delta z_5} - S_5$$

Jacobian becomes:

$$\mathbf{J}_{\text{Neu–Neu}} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 \\ 0 & 0 & 0 & a_5 & b_5 \end{bmatrix}$$

Here: - $b_1 = \partial F_1 / \partial h_1$, $c_1 = \partial F_1 / \partial h_2$ - $a_5 = \partial F_5 / \partial h_4$, $b_5 = \partial F_5 / \partial h_5$

—

Case C: Free–Free (Unit Gradient)

Free drainage implies $\frac{\partial h}{\partial z} = -1$ at boundaries. Similar to Neumann, but gradient is not directly specified as flux.

Residuals look like interior nodes:

$$F_1 = \cdots + \frac{q_{3/2} - q_{1/2}}{\Delta z_1}, \quad F_5 = \cdots + \frac{q_{11/2} - q_{9/2}}{\Delta z_5}$$

Jacobian is:

$$\mathbf{J}_{\text{Free–Free}} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 \\ 0 & 0 & 0 & a_5 & b_5 \end{bmatrix}$$

—

Case D: No-Flow (Zero Gradient)

Imposes $q = 0$ at boundaries, e.g., impermeable layers. Residuals again reflect flux balance, and Jacobian is tridiagonal:

$$\mathbf{J}_{\text{NoFlow}} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 \\ 0 & 0 & 0 & a_5 & b_5 \end{bmatrix}$$

In all non-Dirichlet cases, the system remains fully populated and consistent.

—

Summary of Stencil

The tridiagonal Jacobian entries are derived from:

$$\begin{aligned} a_i &= \frac{\partial q_{i-1/2}}{\partial h_{i-1}} \cdot \frac{1}{\Delta z_i} \\ b_i &= \frac{\partial \theta_i}{\partial h_i} \cdot \frac{1}{\Delta t} + S_{s,i} \cdot \frac{1}{\Delta t} - \frac{\partial q_{i+1/2}}{\partial h_i} \cdot \frac{1}{\Delta z_i} + \frac{\partial q_{i-1/2}}{\partial h_i} \cdot \frac{1}{\Delta z_i} \\ c_i &= -\frac{\partial q_{i+1/2}}{\partial h_{i+1}} \cdot \frac{1}{\Delta z_i} \end{aligned}$$

where all flux derivatives depend on the gradient $\frac{\partial h}{\partial z}$ and the conductivity averaging method (e.g., arithmetic mean).

This structure ensures sparsity and facilitates fast solving via direct or iterative solvers.

2.8 Adaptive Time Stepping

The time step Δt is adjusted at the end of each successful iteration:

$$\Delta t_{\text{new}} = \begin{cases} \min(\Delta t \cdot \beta_{\text{up}}, \Delta t_{\max}) & \text{if } n_{\text{iter}} \leq n_{\text{up}} \\ \max(\Delta t \cdot \beta_{\text{down}}, \Delta t_{\min}) & \text{if } n_{\text{iter}} \geq n_{\text{down}} \\ \Delta t & \text{otherwise} \end{cases}$$

Typical parameters:

- $\beta_{\text{up}} = 1.5$, $\beta_{\text{down}} = 0.5$
- $n_{\text{up}} = 6$, $n_{\text{down}} = 15$
- $\Delta t_{\min}, \Delta t_{\max}$ user-defined

2.9 Mass Balance Evaluation

After each successful step, the model computes mass balance diagnostics:

$$\Delta S = \sum_i \theta_i^{n+1} \Delta z_i - \sum_i \theta_i^n \Delta z_i \tag{14}$$

$$\text{Net Input} = \left(-\bar{q}_{\text{top}} + \bar{q}_{\text{bot}} + \sum_i \bar{S}_i \Delta z_i \right) \Delta t \tag{15}$$

$$\text{Mass Error} = \Delta S - \text{Net Input} \quad [\text{m}] \tag{16}$$

where:

- $\bar{q}$ and $\bar{S}$ are time-averaged over $[t, t + \Delta t]$
- The total ponding volume is included in ΔS if applicable

The cumulative error is tracked and printed if above a threshold (e.g., 1e−6 m).

3 Model Architecture

The model is designed in a modular and extensible fashion, with clearly defined functional roles for each script and folder. This allows users to modify or extend components independently (e.g., use different runoff models, or replace the Richards solver with a multi-dimensional variant).

3.1 Folder Structure

The codebase is organized into the following directories:

- `Main_Function/` – Solver launcher, core integrator, and post-processing driver
- `Numerical_Solver/` – Functions for computing residuals, Jacobians, and time stepping logic
- `Physical_Functions_And_Utilities/` – Soil physics (e.g., Van Genuchten functions), evaporation reduction, mesh generation, boundary logic
- `Forcing/` – Rainfall time series, ETP, SCS-CN hydrology, pollutant models
- `Model_Configurations/` – Scenario definitions (soil layering, BCs, mesh)
- `Visualization_And_Output/` – Plotting, animations, exports (Excel/MP4/PNG)

3.2 Core Functional Flow

At a high level, the simulation is executed in the following sequence:

1. **Initialize Input**
   - Script: `Input_Data.m`
   - Purpose: define mesh, time step settings, soil properties, boundary types, and scenario name
2. **Process Forcing and Hydrologic Inputs**
   - Scripts: `Catchment_Outputs.m`, `Rainfall_ETP_Timeseries.m`
   - Purpose: compute surface fluxes (e.g., infiltration, ET, overflow) for Neumann boundary conditions
3. **Run Richards Solver**
   - Script: `Main_Solver.m`
   - Purpose: solve the nonlinear PDE using Newton–Raphson with adaptive time stepping

4. **Post-Process Results**
    - Scripts: `Post_Processing.m`, `SaveSimulationResults.m`
    - Purpose: create plots, animations, Excel exports, and mass balance reports

3.3 Functional Independence and Extensibility

Each component is functionally independent:

- Soil physics can be swapped (e.g., Brooks–Corey, Campbell) by editing `theta_vgm.m`, `K_vgm.m`
- Runoff forcing can come from SCS-CN, a manual hydrograph, or real data
- Drainage structures (e.g., orifice, spillway) are fully modular and analytical derivatives are exposed

3.4 Parameterization Strategy

All inputs are passed via a unified `params` structure, containing:

- Soil hydraulic properties per layer
- Mesh and discretization info
- Time stepping configuration
- Boundary condition types and values
- Source/sink profiles and pollutant flags

This structure ensures clarity and reproducibility for batch simulations or ensemble modeling.

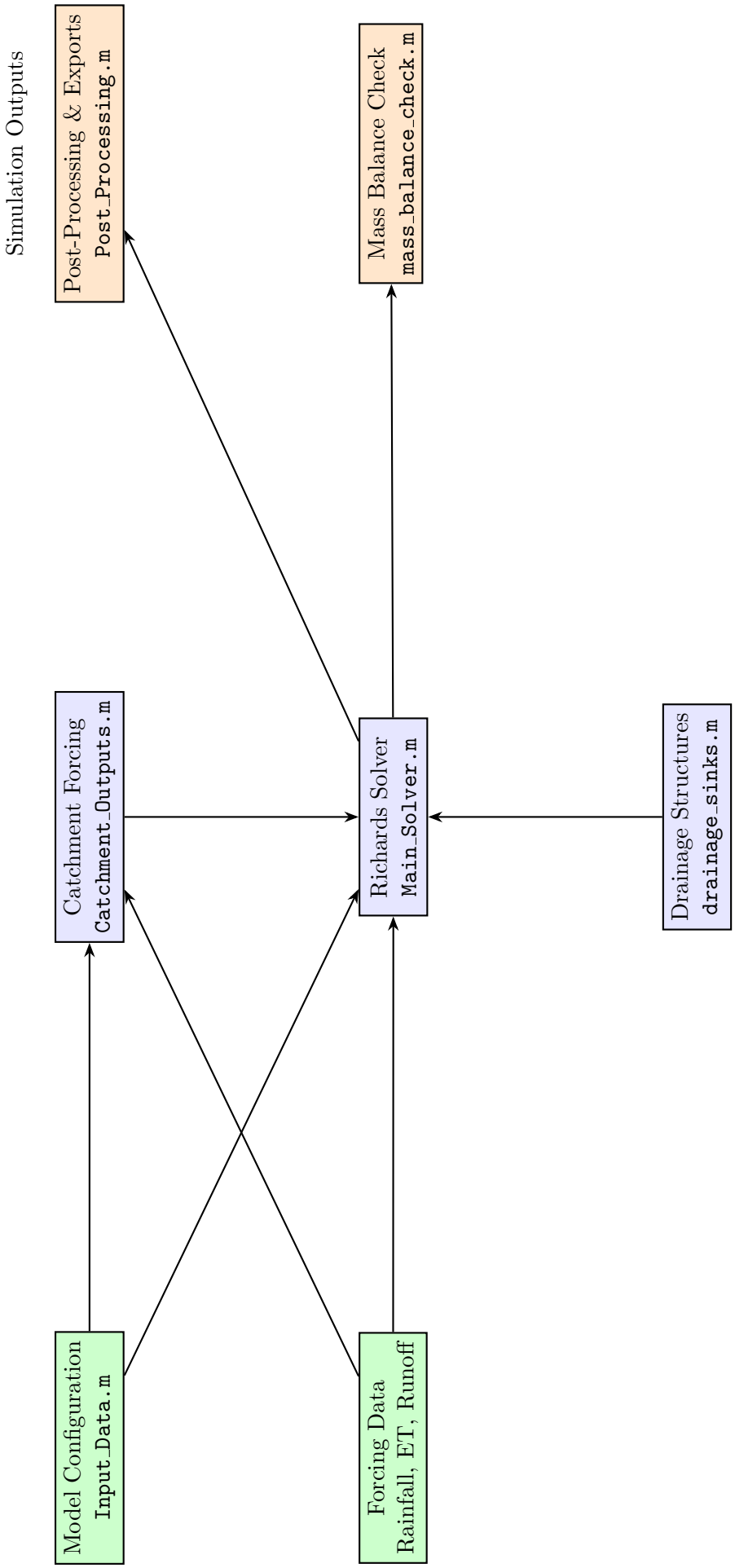

Figure 1: Final architecture showing left-to-right data flow from configuration and forcing through the Richards solver, drainage feedback, and export routines.

4 Input Configuration

The simulation is configured through the script `Input_Data.m`, located in the `Model_Configurations` folder. This script defines the entire simulation scenario by initializing the mesh, time parameters, soil layers, boundary conditions, and file output structure.

All inputs are stored in a single MATLAB structure named `params`, which is passed to the solver.

4.1 Mesh and Domain Setup

The vertical domain extends from depth $z = -L$ [m] (bottom) to $z = 0$ [m] (surface). It is discretized into N_z node-centered control volumes.

Each node i is associated with:

- a vertical position z_i [m], where $z_1 = -L$ and $z_{N_z} = 0$
- a thickness Δz_i [m] representing the vertical span between interfaces

Uniform Mesh

In a uniform grid, each cell has equal thickness:

$$\Delta z_i = \frac{L}{N_z - 1} \tag{17}$$

$$z_i = -L + (i-1) \cdot \Delta z_i, \quad i = 1, \ldots, N_z \tag{18}$$

This is simple to implement but may poorly resolve surface infiltration fronts, particularly when soil moisture gradients are steep near the top boundary.

Refined (Nonlinear) Mesh Toward Surface

To improve resolution near the surface, the model supports a power-law mesh stretching defined by:

$$\eta_i = \left(\frac{i-1}{N_z - 1} \right)^n, \quad n > 1 \quad \text{(refinement exponent)} \tag{19}$$

$$z_i = -L + L \cdot \eta_i \tag{20}$$

Where:

- $\eta_i \in [0, 1]$ is a normalized spacing coordinate
- n controls the degree of refinement:
  - $n = 1$ yields uniform spacing
  - $n > 1$ clusters nodes near the surface ($z = 0$)

The mesh is generated using the utility function:

```
[z, dz] = generate_nonlinear_mesh(L, Nz, n);
```

which returns:

- `z` – a column vector with node elevations (top to bottom)
- `dz` – a vector of corresponding control volume thicknesses

Notes on Mesh Orientation

The model uses a convention where:

- $i = 1$ corresponds to the deepest node (bottom)
- $i = N_z$ corresponds to the surface node
- Δz_i is defined between $z_{i-1/2}$ and $z_{i+1/2}$ for control volume-centered discretization

This orientation is consistent with gravitational drainage and ponding behavior. All spatial derivatives in the residual function are computed using this mesh.

Recommended Settings

For typical infiltration or drainage problems, recommended settings are:

- $N_z = 21$ to 101 nodes
- $n = 1.1$ to 1.5 for surface refinement

4.2 Time Stepping Parameters

The following fields control the simulation time domain:

- `dt` – initial time step size [s]
- `dt_min`, `dt_max` – adaptive limits [s]
- `Tmax` – total simulation time [s]
- `save_interval` – how often to save outputs [s]

4.3 Soil Hydraulic Properties

The domain can be divided into layers, each with unique Van Genuchten and conductivity parameters:

- `theta_r`, `theta_s` – residual/saturated moisture [m^3/m^3]
- `alpha`, `n`, `m` – Van Genuchten shape parameters [1/m], [–]
- `Ks` – saturated hydraulic conductivity [m/s]
- `Ss` – specific storage [1/m]
- `layer_indices` – nodes associated with each soil layer

4.4 Boundary Conditions

Boundary conditions are applied at the top ($i = N_z$) and bottom ($i = 1$) of the column.

Top Boundary:

- `top_bc_type` = 'Dirichlet', 'Neumann', 'Free', or 'NoFlow'
- `top_bc_value` = pressure head [m] or surface flux [m/s]

Bottom Boundary:

- `bottom_bc_type` = 'Dirichlet', 'Neumann', 'Free', or 'NoFlow'
- `bottom_bc_value` = pressure head [m] or flux [m/s]

If Neumann is selected for the top, then rainfall and evapotranspiration must be supplied via a forcing module.

4.5 Source and Sink Terms

The following optional arrays may be defined:

- `source_term` – volumetric source $S(z, t)$ [1/s], e.g., root uptake
- `ponding_depth` – initial ponding head [m], optional

4.6 Drainage Structures

Subsurface or overflow drainage can be enabled:

- `orifice_enabled` – boolean
- `orifice_node` – node index for outflow [–]
- `orifice_coeff` – orifice coefficient [$\sqrt{m}$]
- `spillway_enabled`, `spillway_coeff` – overflow weir flow [$\mathrm{m}^{3/2}/\mathrm{s}$]

The model includes explicit physical formulations for surface overflow and underdrain structures, implemented in `drainage_sinks.m`. These operate on the top node (ponding depth) or any interior node and apply additional sink terms to the residual.

Orifice Flow

The orifice simulates subsurface outflow driven by hydrostatic head at a designated node. It is governed by:

$$Q_{\text{orifice}} = C_o \cdot A_o \cdot \sqrt{2g \cdot H} \tag{21}$$

Where:

- C_o – orifice coefficient (dimensionless, typically 0.6–0.9)
- A_o – orifice area [m^2]
- g – gravitational acceleration [$9.81\,\mathrm{m/s}^2$]
- H – water head above the orifice centerline [m]

In the model, Q_{orifice} is divided by surface area [m^2] to give a flux term [m/s] added to the residual sink term at the orifice node.

Spillway (Weir) Flow

The spillway simulates surface overflow from ponding using a broad-crested weir formulation:

$$Q_{\text{spillway}} = C_w \cdot L \cdot H^{3/2} \tag{22}$$

Where:

- C_w – spillway coefficient [$m^{1/2}/s$]
- L – weir length [m]
- H – ponding depth above spillway crest [m]

Again, the output is converted to a surface flux [m/s] and subtracted from the residual at the top node.

Activation and Capping

The drainage module supports:

- Capped drainage (zero flux if $H < 0$)
- Smooth transitions to/from active states
- Analytical Jacobian derivatives for Newton convergence

These systems can be combined or toggled independently using the flags:

```
params.orifice_enabled = true;
params.spillway_enabled = true;
```

This module allows simulating bioretention cells, overflow chambers, and stormwater outflow controls.

4.7 Initial Conditions

The initial pressure head $h(z, t = 0)$ is usually set to hydrostatic equilibrium (e.g., $\frac{\partial h}{\partial z} = -1$). This is generated analytically from z and the bottom boundary condition.

4.8 Output Paths and Metadata

Each simulation is stored in a unique folder tree:

- `base_output_dir` – root folder for this scenario
- `data_dir` – numerical results (.xlsx)
- `figures_dir` – static plots (.png)
- `animations_dir` – vertical profiles (.mp4)

A simulation log (`Log.txt`) is created summarizing all inputs, solver tolerances, and boundary configurations.

4.9 Simulation Log File

At the end of each simulation, a log file named `Log.txt` is automatically created in the output directory (e.g., `Results_MyScenario/`). This log provides a structured summary of all model settings and is useful for traceability, reproducibility, and debugging.

The log includes the following sections:

- **Simulation Metadata**
  - Scenario name
  - Date and time of execution
  - Output folder paths
- **Domain and Discretization**
  - Number of nodes (`Nz`)
  - Total depth (`L`) [m]
  - Grid refinement factor (if applicable)
- **Time Stepping Settings**
  - Initial time step (`dt`) [s]
  - Minimum and maximum adaptive steps
  - Total simulation time (`Tmax`) [s]
  - Save interval [s]
- **Solver Configuration**
  - Maximum Newton iterations
  - Residual tolerance
  - Jacobian method (finite difference)
  - Line search enabled/disabled
- **Soil Layers and Properties**
  - Number of soil layers
  - For each layer: `theta_r`, `theta_s`, `alpha`, `n`, `Ks`, `Ss`
  - Node indices defining each layer
- **Boundary Conditions**
  - Top and bottom types (Dirichlet, Neumann, etc.)
  - Specified values (e.g., fixed head or flux) [m, m/s]
- **Drainage Structures**
  - Whether orifice and spillway are enabled

  - Node location and coefficients

- **Source Terms**
  - If present, describes source/sink profile
  - Notes any dynamic flux input from runoff module

- **Remarks and Developer Info**
  - Author and version
  - Contact email for support

This file is saved as plain text for compatibility with version control systems and can be regenerated at any time from the `params` struct.

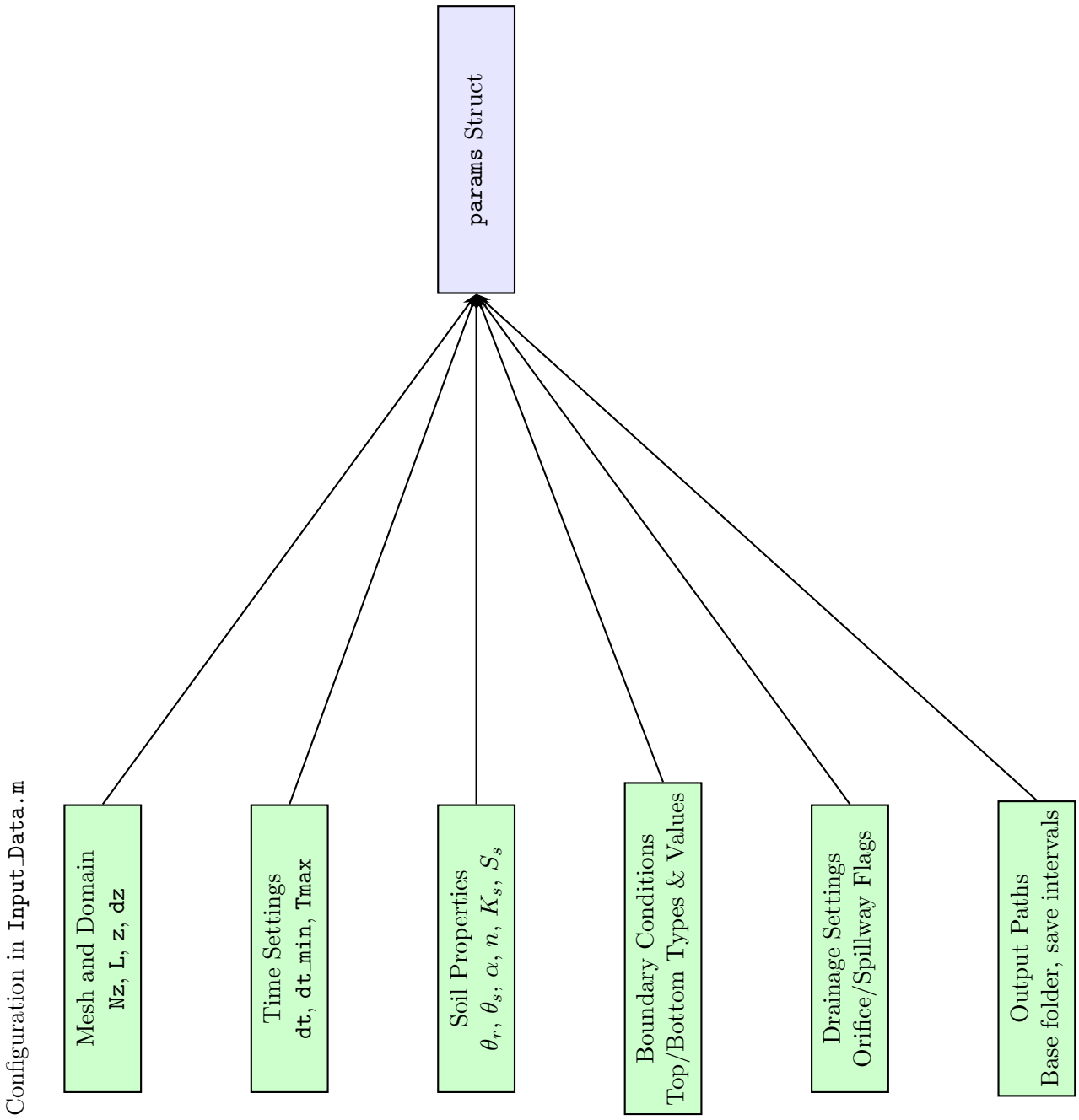

Figure 2: Structure of `Input_Data.m`, showing grouped simulation inputs passed into the `params` struct.

5 Running the Model

Once the input configuration is complete, the model can be launched through the master script `Mixed_Richards_Model_1D.m`. This section explains how to run a simulation, interpret solver behavior, and retrieve outputs.

5.1 Execution Steps

To run the model:

1. Open MATLAB and navigate to the root folder of the codebase.
2. Choose a predefined input script (e.g., `Input_Data.m`) in the `Model_Configurations` folder.
3. Run the launcher:

```
>> Mixed_Richards_Model_1D
```

4. The script will:
   - Load the selected input scenario (.mat or .m)
   - Run the coupled hydrologic model
   - Save all outputs (figures, spreadsheets, animations)
   - Display runtime progress and convergence messages

5.2 Available Test Scenarios

Included in the codebase are multiple test cases:

- `Example1_Infiltration.mat` — surface-driven infiltration
- `Example2_CapillaryRise.mat` — upward water movement
- `Celia1990.mat` — benchmark test against analytical solution
- `PermeablePavement_SanAntonio.mat` — gravel/sand infiltration system
- `Large_Bioretention.mat` — bioretention facility with surface ponding and orifice outflow

You can activate any scenario by setting:

```
sim_name = 'Large_Bioretention';
```

in the launcher script.

5.3 Solver Runtime Feedback

The model currently runs without detailed terminal output. By default, it executes silently unless a convergence failure, time step error, or mass balance instability occurs.

If the solver fails to converge within the allowed number of Newton iterations, the time step Δt is reduced and the step is retried. Warnings may be printed in these cases to alert the user.

To monitor simulation progress or debug, users can manually insert custom print statements inside `Main_Solver.m`, such as:

```
fprintf('Time = %.1f s | t = %.1f s | Newton iterations = %d\n', ...
         current_time, dt, num_iterations);
```

Alternatively, full diagnostic logging is available after the run by checking:

- Residual norms stored in `Post_Processing.m`
- Time series plots of convergence behavior
- Mass balance summaries in the exported Excel file

5.4 Failure Modes and Recovery

The model may reduce Δt and retry if:

- Newton iterations exceed max (e.g., 25)
- Residual norm increases (divergence)
- Mass balance error becomes unstable

After three retries at the minimum time step, the simulation will terminate and suggest the user revise inputs (e.g., wetting front, steep gradients, soil layering).

5.5 Simulation Outputs

Upon completion, the following files are created in a structured folder:

- **SimulationResults.xlsx** – Time series of $h(z,t)$, $\theta(z,t)$, $q(z,t)$, ponding depth
- **Figures/*.png** – Soil profiles, time series plots, flow duration curves
- **Animations/*.mp4** – Dynamic evolution of h, θ, q

- **Log.txt** – Input summary, soil parameters, solver settings

This structure enables integration into automated batch testing, sensitivity studies, or calibration workflows.

6 Output and Diagnostics

At the end of each simulation, the model generates a structured set of outputs organized into folders under the main simulation name (e.g., `Results_Large_Bioretention/`).

6.1 Output File Structure

The output directory includes:

- `Data/SimulationResults.xlsx` – full time series of all primary variables
- `Figures/*.png` – static plots of head, moisture, flux, and wetting front
- `Figures/Animations/*.mp4` – video animations of $h(z,t)$, $\theta(z,t)$, and $q(z,t)$
- `Figures/Diagnostics/*.png` – flow duration curves and convergence analysis
- `Log.txt` – run metadata and configuration summary
- `SimulationInfo.txt` – formatted simulation parameters for archiving

These outputs are created automatically and organized into subfolders by type.

6.2 Time Series and Profiles

The main file `SimulationResults.xlsx` includes:

- **Head (h)** – pressure head at each node [m]
- **Moisture (θ)** – volumetric water content [m^3/m^3]
- **Flux (q)** – Darcy flux at each interface [m/s]
- **Surface Variables** – ponding depth, infiltration rate, overflow

Each sheet includes time in seconds, minutes, and days for easy plotting.

6.3 Mass Balance Diagnostics

Mass conservation is evaluated at every time step by comparing the change in total water storage to the net inflow and outflow across the system. The total water storage includes both:

- Subsurface storage in the soil column: $\sum_i \theta_i \Delta z_i$
- Surface ponding volume (if any): h_{ponding}

The mass balance is computed as:

$$\Delta S = \left[\sum_i \theta_i^{n+1} \Delta z_i + h_{\text{ponding}}^{n+1} \right] - \left[\sum_i \theta_i^{n} \Delta z_i + h_{\text{ponding}}^{n} \right] \quad [\text{m}] \tag{23}$$

$$\text{Net Inflow} = \left(-\bar{q}_{\text{top}} + \bar{q}_{\text{bot}} + \sum_i \bar{S}_i \Delta z_i \right) \cdot \Delta t \quad [\text{m}] \tag{24}$$

$$\text{Mass Error} = \Delta S - \text{Net Inflow} \tag{25}$$

Here:

- h_{ponding} is the surface water depth (if infiltration is limited)
- $\bar{q}_{\text{top}}, \bar{q}_{\text{bot}}$ are average top/bottom fluxes over the time step
- $\bar{S}_i$ is the average source/sink term in cell i

If the absolute value of the mass error exceeds a small threshold (e.g., 1e−6 m), the value is recorded and optionally flagged. The cumulative mass error over the entire simulation is also saved.

6.4 Visualization Tools

The model provides high-quality figures using the scripts:

- `Post_Processing.m` – main plotting interface
- `plot_soil_profiles.m` – head, moisture, and flux vs. depth at selected times
- `generate_richards_animation.m` – creates MP4 animations over time
- `plot_flow_duration_curves.m` – upward/downward FDCs

All figures are generated in vector format (PDF/PNG) using LaTeX fonts and saved to the `Figures` folder.

6.5 Wetting Front and Convergence Analysis

The wetting front movement can be tracked automatically and plotted over time. Users may also analyze:

- Convergence rate of Newton iterations
- Sensitivity of solution to time step changes
- Infiltration capacity vs. applied rainfall rate

These diagnostics are useful for identifying physical transitions (e.g., saturation onset, overflow) and numerical instabilities.

7 Validation and Benchmarks

The model has been tested using a suite of benchmark scenarios and reference cases that validate both numerical accuracy and physical consistency. These tests help confirm that the mixed-form Richards solver behaves as expected across a range of soil types, boundary conditions, and flux regimes.

7.1 Analytical and Semi-Analytical Tests

Celia et al. (1990) Benchmark

The scenario reproduces the classic infiltration benchmark described in:

> Celia, M. A., Bouloutas, E. T., & Zarba, R. L. (1990). A general mass-conservative numerical solution for the unsaturated flow equation. *Water Resources Research*, 26(7), 1483–1496.

Key properties:

- Soil: fine sand
- Initial condition: hydrostatic, with water table at the bottom
- Top BC: constant pressure head
- Bottom BC: free drainage

The model reproduces the benchmark infiltration front and cumulative infiltration with high fidelity. Numerical outputs can be compared against the original figures in Celia et al. (1990).

Gravity Drainage (Pure Richards)

A vertical column is initialized with saturated conditions and allowed to drain freely. The top boundary is set as atmospheric (free drainage), and the bottom as zero pressure head.

The expected behavior is a gradual decline in capillary tension and convergence toward equilibrium. This test verifies that:

- No unphysical oscillations occur
- Time step adaptation handles stiff transitions
- Mass conservation remains stable

7.2 Functional Stress Tests

Ponding and Infiltration Capacity

The model is exposed to a high-intensity rainfall pulse exceeding K_s to test:

- Surface ponding and transition logic
- Overflow activation (if spillway is enabled)
- Mass balance robustness under near-saturation

Drainage and Orifice Activation

A bioretention test case includes:

- Evapotranspiration extraction
- Orifice and/or spillway routing
- Transitions between unsaturated and saturated conditions

These tests verify that drainage terms activate only under physical criteria and that stability is maintained.

7.3 Comparative Behavior Checks

The following properties have been verified:

- Saturated flux matches K_s at steady state

- Infiltration front shape aligns with Green-Ampt behavior at early times
- Mesh refinement improves resolution but preserves mass balance
- Time step size affects iteration count, but not final moisture profile

7.4 Suggested Validation Strategy for New Users

Users are encouraged to validate their own scenarios using:

- Convergence tests with decreasing Δz and Δt
- Comparison to analytical drainage or infiltration limits
- Mass balance error below 1e−6 m over full simulation
- Visual inspection of $h(z,t)$, $\theta(z,t)$, and $q(z,t)$ evolution

These checks help ensure that the model is well-posed and numerically robust under the user's parameter set.

8 Case Studies and Applications

The following case studies demonstrate the model's applicability to real-world hydrologic and hydraulic systems. They combine vadose zone dynamics with engineered boundary conditions such as ponding, overflow, and pollutant transport.

8.1 Infiltration into Homogeneous Soil

File: `Example1_Infiltration.mat`

- Soil: loamy sand, uniform profile
- Top BC: constant rainfall pulse (Neumann)
- Bottom BC: free drainage
- Objective: test unsaturated infiltration and deep percolation

This scenario highlights the wetting front propagation and soil saturation development under a short-duration rainfall event.

8.2 Capillary Rise from Shallow Water Table

File: `Example2_CapillaryRise.mat`

- Soil: fine sand
- Initial Condition: dry upper profile
- Bottom BC: saturated pressure head (Dirichlet)
- Objective: simulate upward movement of moisture into vadose zone

The case demonstrates capillarity and moisture redistribution without rainfall.

8.3 Celia et al. (1990) Benchmark

File: `Celia1990.mat`

Described in Section 8, this test case provides a reference for model accuracy under known boundary conditions and exact solution structure.

8.4 Permeable Pavement with Gravel Subbase

File: `PermeablePavement_SanAntonio.mat`

- Soil: gravel + sand layers
- Top BC: rainfall hydrograph (Neumann), real data
- Drainage: orifice at base with capped flow
- Objective: simulate stormwater infiltration + delayed exfiltration

This scenario highlights rapid percolation in coarse media and delayed drainage.

8.5 Bioretention Facility with Overflow

File: `Large_Bioretention.mat`

- Layered soil: media + storage layer + subdrain
- Top BC: rainfall + ET (Neumann), dynamic forcing
- Drainage: orifice at depth + spillway at surface
- Objective: simulate overflow events and long-term drainage performance

This case is ideal for testing all model features: ponding, ET, underdrain flow, and saturation switching.

8.6 Custom Urban Catchment

User-defined via `Rainfall_ETP_Discharge_Timeseries.m` and hydrograph inputs.

- Load rainfall, PET, and observed runoff
- Derive surface Neumann flux via SCS-CN or manual hydrograph
- Run infiltration simulations for urban soils

Useful for data-driven applications, including calibration and parameter estimation.

8.7 Suggested Modifications for New Applications

The model is flexible enough to simulate:

- Shallow groundwater recharge or drying fronts
- Lysimeter column experiments with controlled drainage
- Vadose zone pollutant breakthrough and retention
- Multi-event rainfall sequences over months or years

Modifications to inputs (e.g., layered soil profile, overflow elevations, or hydrograph forcing) are sufficient to adapt to most new use cases.

9 Troubleshooting and Common Issues

This section summarizes typical problems that may arise during model execution and how to resolve them. The model is numerically robust, but strong nonlinearities, sharp boundary transitions, or incorrect input configurations may still cause instability or failure.

9.1 Convergence Failure in Newton Iteration

Symptoms:

- Maximum number of Newton iterations reached
- Residual norm increases instead of decreasing
- Time step is reduced repeatedly without success

Causes and Fixes:

- Time step Δt too large → reduce initial `dt` and `dt_max`
- Sharp transitions in boundary conditions → smooth rainfall forcing
- Improper initial condition (e.g., hydrostatic far from expected state) → recompute `h_init`
- Near-saturation leading to stiffness → consider finer vertical mesh or stronger line search damping

9.2 Mass Balance Error Accumulation

Symptoms:

- Cumulative mass error exceeds 1e−5 m
- Sudden spikes in mass error during intense infiltration or drainage

Causes and Fixes:

- Ponding dynamics not captured accurately → verify top Neumann logic and flux interpolation
- Drainage thresholds incorrectly triggered → check orifice/spillway depth logic
- Adaptive Δt jumps over critical transitions → tighten residual tolerance

9.3 Unphysical Results (Negative θ, Oversaturation)

Symptoms:

- Water content exceeds θ_s or drops below θ_r
- Large oscillations near boundaries

Causes and Fixes:

- Incorrect Van Genuchten parameters → cross-check input values and units
- Floating-point instability at low gradients → enforce min/max θ in post-processing
- Spurious fluxes across large Δz transitions → smooth the mesh or refine upper layers

9.4 No Overflow or Drainage Despite Ponding

Symptoms:

- Ponding depth increases but orifice/spillway does not activate
- No outflow shown in diagnostics

Causes and Fixes:

- Drainage flags not enabled → ensure `params.orifice_enabled = true`
- Coefficients set too low → increase `orifice_coeff`, `spillway_coeff`
- Ponding depth below threshold → check drainage elevation vs. actual h_{ponding}

9.5 Solver Terminates Without Output

Symptoms:

- No Excel or figures created
- Solver exits without warning

Causes and Fixes:

- Fatal convergence failure (after max retries) → review log, reduce `dt`
- Path issues in `Input_Data.m` → confirm `base_output_dir` exists
- Incomplete parameter struct → check all required `params` fields are defined

9.6 Recommended Debugging Strategy

- Start with a simple, well-behaved scenario (e.g., infiltration into uniform loamy sand)
- Disable drainage and forcing modules to isolate solver behavior
- Use short simulation time and small Δt to observe transient behavior
- Inspect residuals and intermediate plots using built-in figures
- Compare cumulative mass error at each step

If issues persist, users are encouraged to check:

- `mass_balance_check.m` results
- Final `Log.txt` for parameter summary
- Matrix condition numbers (if adding custom Jacobians)

10 Frequently Asked Questions (FAQ)

This section compiles the most common questions raised by users and developers when working with the model.

General Model Capabilities

Q1: Does the model simulate both unsaturated and saturated conditions?

Yes. The mixed-form Richards equation allows continuous simulation from dry to saturated soil states, including positive pressure heads (ponding).

Q2: Can the model represent capillary rise?

Yes. By setting a Dirichlet boundary condition at the bottom with fixed water table pressure, the model can simulate upward moisture movement.

Q3: Can I simulate infiltration under time-varying rainfall?

Yes. Use the `Rainfall_ETP_Timeseries.m` or `Catchment_Outputs.m` scripts to apply realistic, time-varying Neumann fluxes.

Q4: Is it possible to model surface ponding and overflow?

Yes. When rainfall exceeds infiltration capacity, water accumulates at the surface as ponding depth. Overflow can be routed via a spillway if enabled.

Q5: Can I simulate artificial underdrain structures like an orifice?

Yes. The model allows you to place an orifice at any node, and computes drainage based on hydrostatic head using Bernoulli-based flow formulas.

Numerical and Solver Behavior

Q6: What should I do if Newton-Raphson fails to converge?

First, reduce the initial `dt`. If issues persist, enable line search, smooth boundary inputs, or refine the mesh near steep gradients.

Q7: What tolerance does the solver use to stop iterations?

The residual norm $\|\mathbf{F}(\mathbf{h})\|$ must fall below a user-defined tolerance (typically 1e−6 m).

Q8: Does the solver support adaptive time stepping?

Yes. Time steps grow or shrink based on how many iterations are needed. This improves efficiency and handles stiff gradients.

Q9: What is the effect of the mesh refinement exponent?

Higher exponents (e.g., $n = 3$) cluster nodes near the surface, which improves resolution of steep wetting fronts but increases computation time.

Q10: Can I inspect the Jacobian matrix?

Yes. The Jacobian is assembled numerically via finite differences and is stored as a sparse matrix. You can output it for inspection.

Boundary Conditions and Configuration

Q11: What boundary types are supported?

Dirichlet, Neumann, Free Drainage, and No-Flow are supported at both the top and bottom boundaries.

Q12: What happens if I use Neumann BC but the soil saturates?

The model automatically adjusts surface flux and accumulates ponding, ensuring physical realism when infiltration capacity is exceeded.

Q13: How do I apply time-varying evapotranspiration?

Use the forcing scripts to pass ETP as a negative flux at the surface. The Feddes function can reduce it dynamically based on soil suction.

Q14: How is rainfall input converted to infiltration?

Rainfall becomes a top Neumann flux. The model compares rainfall to infiltration capacity, subtracts surface losses, and manages ponding.

Q15: Can I apply a hydrograph instead of computing runoff?

Yes. You can bypass SCS-CN by loading your own hydrograph and converting it to a Neumann boundary condition.

Outputs and Interpretation

Q16: What files are saved after simulation?

Results include an Excel file of time series, PNG figures, MP4 animations, and plain-text logs of model parameters and simulation summaries.

Q17: How do I check if the simulation is accurate?

Ensure mass balance error remains below 1e−6 m, and visually verify that $h(z, t)$ and

$\theta(z,t)$ evolve smoothly.

Q18: Can I plot the wetting front movement?

Yes. The model tracks the depth at which θ increases sharply and can produce a figure showing wetting front vs. time.

Q19: Where are ponding depths and overflows stored?

They are stored in the surface output arrays in `SimulationResults.xlsx`, and plotted in the default time series figure.

Q20: Can I compare infiltration vs. runoff for a storm?

Yes. Use the forcing module to generate both rainfall and runoff time series, then compare them against infiltration and overflow outputs.

Appendix A: Typical Van Genuchten Parameters by Soil Texture

The following table summarizes approximate values of Van Genuchten parameters for standard USDA soil texture classes. These values are adapted from Carsel and Parrish (1988) and are widely used in hydrologic modeling.

Soil Texture	θ_r	θ_s	α [1/m]	n [–]	K_s [m/s]	S_s [1/m]
Sand	0.045	0.43	14.5	2.68	8.25×10^{-5}	1×10^{-5}
Loamy Sand	0.057	0.41	12.4	2.28	2.50×10^{-5}	1×10^{-5}
Sandy Loam	0.065	0.41	7.5	1.89	1.06×10^{-5}	1×10^{-5}
Loam	0.078	0.43	3.6	1.56	2.89×10^{-6}	1×10^{-5}
Silt Loam	0.067	0.45	2.0	1.41	6.25×10^{-7}	1×10^{-5}
Sandy Clay Loam	0.100	0.39	4.4	1.48	3.75×10^{-6}	1×10^{-5}
Clay Loam	0.095	0.41	1.9	1.31	1.32×10^{-6}	1×10^{-5}
Silty Clay Loam	0.089	0.43	1.3	1.23	8.75×10^{-7}	1×10^{-5}
Sandy Clay	0.100	0.38	2.7	1.23	1.32×10^{-6}	1×10^{-5}
Silty Clay	0.070	0.36	0.8	1.09	4.58×10^{-7}	1×10^{-5}
Clay	0.068	0.38	0.9	1.09	2.45×10^{-7}	1×10^{-5}

Table 2: Typical Van Genuchten–Mualem parameters and specific storage for USDA soil classes.

Appendix B: Dimensional Analysis of Model Variables

The following table summarizes the dimensions and SI units of all primary variables used in the model.

Variable	Description	SI Units / Dimensions
h	Pressure head	[L] (m)
θ	Volumetric water content	[–] (m^3/m^3)
$K(h)$	Hydraulic conductivity	[L T^{-1}] (m/s)
q	Darcy flux	[L T^{-1}] (m/s)
z	Vertical coordinate (upward)	[L] (m)
S_s	Specific storage	[L^{-1}] (1/m)
$S(z,t)$	Source/sink term	[T^{-1}] (1/s)
$C(h)$	Specific moisture capacity	[–] (1/m)
t	Time	[T] (s)
Δz	Cell thickness	[L] (m)
Δt	Time step	[T] (s)
α	VG inverse air-entry pressure	[L^{-1}] (1/m)
n, m	VG shape parameters	[–]
θ_r	Residual moisture content	[–] (m^3/m^3)
θ_s	Saturated moisture content	[–] (m^3/m^3)
K_s	Saturated conductivity	[L T^{-1}] (m/s)
H	Ponding depth / hydrostatic head	[L] (m)
Q	Volumetric flow rate (orifice, weir)	[L^3 T^{-1}] (m^3/s)
C_o	Orifice coefficient	[–]
C_w	Weir (spillway) coefficient	[$L^{1/2}$ T^{-1}] ($m^{1/2}$/s)
L_{weir}	Weir crest length	[L] (m)
A_o	Orifice area	[L^2] (m^2)
g	Gravitational acceleration	[L T^{-2}] (m/s^2)

Table 3: Dimensional summary of key variables in the mixed-form Richards model and drainage modules.

Appendix D: Numerical Algorithm Pseudocode with Function Mapping

This appendix outlines the complete solver algorithm and maps each step to its corresponding MATLAB function.

Main Time Loop — `Main_Solver.m`

```
initialize t = 0
initialize h = h_init from hydrostatic profile
initialize  = theta_vgm(h)        % theta_vgm.m
initialize q = zeros(Nz+1, 1)

while t < Tmax:

    % 1. Get top/bottom boundary fluxes
    [q_top, ponding_depth] = get_boundary_values(...)   % get_boundary_values.m
```

```
    q_bot = derived from BC

    % 2. Catchment flux or hydrograph input
    q_forcing = Catchment_Outputs(...)                    % Catchment_Outputs.m

    % 3. Source/sink terms (e.g., ET, root uptake)
    S = defined in params or derived from forcing

    % 4. Solve Richards equation
    [h_new, q_new, _new] = NewtonSolver(h, , q, S, t) % Newton loop (below)

    % 5. Update ponding volume
    ponding_depth = max(ponding_depth + overflow input - infiltration, 0)

    % 6. Drainage structures
    [q_drainage] = drainage_sinks(...)                    % drainage_sinks.m
    update residual at orifice/spillway nodes

    % 7. Mass balance check
    mass_error = mass_balance_check(...)                  % mass_balance_check.m

    % 8. Save data and increment time
    if mod(t, save_interval) == 0:
        save outputs: h, , q, ponding, overflow        % SaveSimulationResults.m

    adapt t based on convergence history
    t = t + t
```

Newton Solver — internal to `Main_Solver.m`

```
initialize h_k = h_old
iter = 0

while norm(F) > tolerance and iter < max_iter:

    % Evaluate water content and K
    _k = theta_vgm(h_k)                  % theta_vgm.m
    K_k = K_vgm(h_k)                      % K_vgm.m

    % Compute residual vector
    F = compute_residual(h_k, h_old, _k, _old, K_k, S, t)  % compute_residual.m

    % Compute Jacobian
    J = compute_jacobian(h_k, h_old, _k, _old, K_k, S, t)  % compute_jacobian.m

    % Newton update with line search
    h = -J \ F
```

```
    h_k = h_k +  * h
    iter = iter + 1

    optionally: apply line search logic (user-defined)
```

Post-Processing — `Post_Processing.m`

```
plot time series of:
    - ponding depth, overflow, infiltration
    - head and moisture at selected nodes
    - surface flux vs. time

plot vertical profiles:
    - h(z,t), (z,t), q(z,t) at key times
    - wetting front depth vs. time

generate animations:
    - generate_richards_animation.m

compute and plot flow duration curves:
    - plot_flow_duration_curves.m
```

Input Handling

- `Input_Data.m` – defines mesh, soil properties, BCs, drainage flags
- `generate_nonlinear_mesh.m` – builds stretched grid
- `Rainfall_ETP_Timeseries.m` – processes rainfall and ET forcing
- `SCS_Hydrologic_Model.m` – computes runoff via SCS-CN
- `Buildup_Washoff_Model.m` – simulates pollutant dynamics (optional)
- `SaveSimulationResults.m` – saves data to Excel

Appendix D: Numerical Algorithm Pseudocode with Function Mapping

This appendix outlines the complete solver algorithm and maps each step to its corresponding MATLAB function.

Main Time Loop — Main_Solver.m

```
initialize t = 0
initialize h = h_init from hydrostatic profile
initialize  = theta_vgm(h)        % theta_vgm.m
initialize q = zeros(Nz+1, 1)

while t < Tmax:

    % 1. Get top/bottom boundary fluxes
    [q_top, ponding_depth] = get_boundary_values(...)   % get_boundary_values.m
    q_bot = derived from BC

    % 2. Catchment flux or hydrograph input
    q_forcing = Catchment_Outputs(...)                  % Catchment_Outputs.m

    % 3. Source/sink terms (e.g., ET, root uptake)
    S = defined in params or derived from forcing

    % 4. Solve Richards equation
    [h_new, q_new, _new] = NewtonSolver(h, , q, S, t) % Newton loop (below)

    % 5. Update ponding volume
    ponding_depth = max(ponding_depth + overflow input - infiltration, 0)

    % 6. Drainage structures
    [q_drainage] = drainage_sinks(...)                  % drainage_sinks.m
    update residual at orifice/spillway nodes

    % 7. Mass balance check
    mass_error = mass_balance_check(...)                % mass_balance_check.m

    % 8. Save data and increment time
    if mod(t, save_interval) == 0:
        save outputs: h, , q, ponding, overflow        % SaveSimulationResults.m

    adapt t based on convergence history
    t = t + t
```

Newton Solver — internal to Main_Solver.m

```
initialize h_k = h_old
iter = 0

while norm(F) > tolerance and iter < max_iter:

    % Evaluate water content and K
```

```
    _k = theta_vgm(h_k)                  % theta_vgm.m
    K_k = K_vgm(h_k)                      % K_vgm.m

    % Compute residual vector
    F = compute_residual(h_k, h_old, _k, _old, K_k, S, t)  % compute_residual.m

    % Compute Jacobian
    J = compute_jacobian(h_k, h_old, _k, _old, K_k, S, t)  % compute_jacobian.m

    % Newton update with line search
    h = -J \ F
    h_k = h_k +  * h
    iter = iter + 1

    optionally: apply line search logic (user-defined)
```

Post-Processing — Post_Processing.m

```
plot time series of:
    - ponding depth, overflow, infiltration
    - head and moisture at selected nodes
    - surface flux vs. time

plot vertical profiles:
    - h(z,t), (z,t), q(z,t) at key times
    - wetting front depth vs. time

generate animations:
    - generate_richards_animation.m

compute and plot flow duration curves:
    - plot_flow_duration_curves.m
```

Input Handling

- Input_Data.m – defines mesh, soil properties, BCs, drainage flags
- generate_nonlinear_mesh.m – builds stretched grid
- Rainfall_ETP_Timeseries.m – processes rainfall and ET forcing
- SCS_Hydrologic_Model.m – computes runoff via SCS-CN
- Buildup_Washoff_Model.m – simulates pollutant dynamics (optional)
- SaveSimulationResults.m – saves data to Excel

How to Cite This Model

If you use this model in your research or applications, please cite:

> Marcus N. Gomes Jr. (2025).
> *A Coupled 1D Mixed-Form Richards and Catchment-Scale Hydrologic Model for Simulating Vadose Zone Dynamics in Engineered Drainage Systems.*
> GitHub Repository: https://github.com/[your-repo-here]
> DOI (if available): 10.5281/zenodo.xxxxxxx

For academic publications using the model, please cite the corresponding journal article or technical report as well.

License and Reuse Terms

This model is distributed under the MIT License. You are free to use, modify, and redistribute the code and documentation provided that:

- The original authors are credited
- The license file is included with any redistributed version
- Any derived publications acknowledge the original model

MIT License Summary:

> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, subject to the following conditions...

The full license text is included in the repository.