Contents

TerraPercolatio — User Manual

Finite-element groundwater seepage analysis and flow nets for Mathematica.

Version 1.5.0 · Requires Mathematica 13.0+ (uses `NDSolve`FEM``) · MIT License · Author: Malcolm Woodruff

---

1. Description

TerraPercolatio solves steady-state groundwater seepage by the finite element method and draws true flow nets. It handles confined and unconfined problems in plane (2D) or axisymmetric geometry, with multi-layer, anisotropic soils. For unconfined problems it locates the phreatic surface and seepage faces iteratively, applies rainfall/surface infiltration through an unsaturated zone, and produces pore-pressure fields — including capped suction — ready for limit-equilibrium slope-stability analysis. Equipotentials come from the head field; flow lines come from the exact conjugate (stream-function) problem, so the flow net is correct in heterogeneous, anisotropic ground.

One-line summary: *2D and axisymmetric FEM seepage, flow nets, phreatic surfaces, rainfall, and pore-pressure output for geotechnical analysis.*

Typical applications: flow under and around sheet-pile walls and cofferdams, dewatered excavations, seepage through earth and zoned dams, seepage and water-table drawdown in slopes, radial flow to pumped wells, and the design of drains and wellpoint lines.

---

2. Installation

Install the paclet and load the package:

```
PacletInstall["/path/to/MalcolmWoodruff__TerraPercolatio-1.4.0.paclet"]
Needs["TerraPercolatio`"]
```

Once published on the Wolfram Paclet Repository:

```
PacletInstall["MalcolmWoodruff/TerraPercolatio"]
Needs["TerraPercolatio`"]
```

If you reload the package after an update, quit the kernel first (`Quit[]`); `Needs` will not re-read a context that is already loaded, and stale definitions can shadow the new ones.

Open the worked-example notebook and these notes from the installed paclet:

```
NotebookOpen @ PacletObject["MalcolmWoodruff/TerraPercolatio"]["AssetLocation", "DemoNotebook"]
SystemOpen  @ PacletObject["MalcolmWoodruff/TerraPercolatio"]["AssetLocation", "README"]
```

---

3. Method and theory

The solver treats saturated flow governed by Darcy's law and continuity, giving, for steady state,

$\nabla\cdot(\mathrm{K}\ \nabla\mathrm{h}) = 0$, with $\mathrm{K} = \mathrm{diag}(\mathrm{kx}, \mathrm{ky})$

piecewise constant per layer, where h is total head (elevation + pressure head). This is discretised with the Mathematica finite element method on an unstructured mesh. Boundary conditions are Dirichlet (prescribed head) on water boundaries and natural no-flow elsewhere; seepage faces and free surfaces are handled specially.

Flow nets by the conjugate problem. The stream function ψ satisfies the dual equation $\nabla\cdot(\tilde{K}\ \nabla\psi) = 0$ with $\tilde{K} = \mathrm{diag}(1/\mathrm{ky}, 1/\mathrm{kx})$ and the boundary types swapped (head boundaries become natural, streamline boundaries become Dirichlet). Solving this exact dual gives streamlines that refract correctly across layer interfaces and in anisotropic ground, so contours of equal ψ-increment bound channels of equal discharge.

Unconfined free surface. For dams and slopes the phreatic surface is unknown. TerraPercolatio meshes the saturated region below a trial surface, solves for head, and relocates the surface to where pressure head is zero (h = y), iterating to convergence. Below the exit point on the downstream face the boundary becomes a seepage face (h = y). The water table is capped at the ground surface; the gap between them is the unsaturated zone.

Axisymmetric flow. With `"Axisymmetric" -> True` the coordinate x is the radial coordinate r; the weak form is weighted by r, discharge Q is returned as the full flow in m^3/s, and ψ is the Stokes stream function (equal $\Delta\psi$ still bounds equal-flow tubes). `SeepageWell` wraps this with the drawdown-cone iteration.

Rainfall and the unsaturated zone. Surface infiltration (`"Rainfall"` in mm/day, or `"Recharge"` in m/s) is applied as a flux across the phreatic surface, assuming vertical unit-gradient percolation through the unsaturated zone. It raises the water table and pore pressures. Above the water table, pore pressure is taken as capped hydrostatic suction; below it, the FEM value is used. Rainfall can only infiltrate as fast as the soil's vertical conductivity ky allows — if the applied rate exceeds ky the water table mounds unrealistically (in reality the surplus runs off), so keep rainfall below ky.

Assumptions and scope. Steady state; rigid soil; Darcy (laminar) flow; the unsaturated zone is represented by capped hydrostatic suction rather than a full Richards-equation solution; free surfaces are assumed single-valued (x-monotone). These are the standard assumptions of flow-net and limit-equilibrium practice.

---

4. Quick start

Confined flow (uniform strip, left-to-right):

```
Needs["TerraPercolatio`"]
layers  = {<|"Polygon" -> {{0,0},{20,0},{20,10},{0,10}}, "kx" -> 1.*^-5, "ky" -> 1.*^-5|>};
headBCs = {{12., Function[{x,y}, x <= 0.001]},     (* h = 12 on the left  *)
           {10., Function[{x,y}, x >= 19.999]}};   (* h = 10 on the right *)
streams = {Function[{x,y}, y >= 9.999],            (* top streamline    *)
           Function[{x,y}, y <= 0.001]};           (* bottom streamline *)
sol = SeepageSolve[layers, headBCs, streams];
FlowNet[sol]
sol["Q"]        (* discharge, m^3/s per metre run *)
```

Unconfined slope with rainfall, and pore pressures for stability:

```
spec = <|"Profile" -> {{0,-2},{75,-2},{75,0},{50,0},{30,10},{0,10}},
         "UpstreamLevel" -> 8., "DownstreamLevel" -> 0.,
         "UpstreamFace" -> {{0,-2},{0,10}},
         "DownstreamFace" -> {{75,-2},{75,0},{50,0},{30,10}},
         "kx" -> 1.*^-6, "ky" -> 1.*^-7|>;
sol = SeepageUnconfined[spec, "Rainfall" -> 5.];
SeepagePlot[sol, "PorePressure", "SuctionCap" -> 1.]
u = PorePressureField[sol];      (* u[x,y] in kPa, for export *)
```

---

5. Function reference

SeepageSolve

`SeepageSolve[layers, headBCs, streamBoundaries]` — confined solver.

- **layers** — list of `<|"Polygon" -> {{x,y}..}, "kx" -> kx, "ky" -> ky|>` (or `{polygon, kx, ky}`). Adjacent layer polygons must share vertices along common interfaces; layers are searched in order (first match wins).
- **headBCs** — `{{value, pred}, ...}`, where `pred` is `(x,y) -> True` on that boundary and `value` is a number or a function of `(x, y)`.
- **streamBoundaries** — `{predA, predB}`, the two bounding impermeable streamlines (e.g. a cutoff wall and the base). For more than two groups give `"PsiValues"`.

Key options: `"Axisymmetric"` (False), `"SeepageFaces"` ({}), `"RechargeBC"` (None), `"MeshSize"` (Automatic max element area), `"MeshRefinementFunction"`, `"PsiValues"`, `"Mesh"`, `"Name"`.

Returns an association (see §6). Example: see Quick start.

SeepageUnconfined

`SeepageUnconfined[spec]` — unconfined solver that locates the phreatic surface.

`spec` keys: `"Profile"` (counter-clockwise cross-section polygon), `"UpstreamLevel"` (Hu), `"DownstreamLevel"` (Hd), and either `"Layers"` (as in SeepageSolve) or homogeneous `"kx"`, `"ky"`. For non-trivial profiles give `"UpstreamFace"` and `"DownstreamFace"` as polylines ordered from the base upward. Flow is left → right; the base (minimum y) is impermeable.

Key options: `"Rainfall"` (0, mm/day) or `"Recharge"` (0, m/s) — surface infiltration; `"UpstreamNoFlow"` (False) — make the upstream face a mid-spacing no-flow divide (drain-spacing design); `"Points"` (25) free-surface stations; `"Relaxation"` (0.65); `"MaxIterations"` (100); `"Tolerance"` (Automatic); `"Monitor"` (False) prints the iteration history; `"MeshSize"`; `"Name"`.

Additional return keys: "PhreaticSurface", "ExitPoint", "SaturatedRegion", "GroundSurface", "UnsaturatedZone", "UnsaturatedThickness" ({min,max} gap), "WaterTableDaylights", "Rainfall_mmday", "Converged", "Iterations".

SeepageWell

SeepageWell[spec] — axisymmetric unconfined flow to a pumped well; locates the drawdown cone and the seepage face on the screen.

spec keys: "WellRadius" (rw), "OuterRadius" (R, radius of influence), "FarWaterTable" (H above the base), "WellWaterLevel" (hw), "kr", "kz". Coordinates are (r, z) with the base at z = 0. Q is returned in m^3/s. The association also carries "DupuitQ" (the Dupuit–Thiem estimate) for comparison.

Options: "Points" (30), "Relaxation", "MaxIterations", "Tolerance", "Monitor", "MeshSize", "Name".

FlowNet

FlowNet[sol] — draws the flow net (equipotentials + flow lines) and reports Q.

Options: "PotentialDrops" (10) number of head drops Nd; "FlowChannels" (Automatic, chosen so the net is square in "SquareNetLayer"); "WaterLevels" — list of {level, {x1, x2}} drawing free-water surfaces (with the (WT) symbol), including standing water above ground; plus styling and any Graphics options. The phreatic surface and unsaturated zone are shown automatically for unconfined solutions.

SeepagePlot

SeepagePlot[sol, field] — filled contour map of a field. field is "Head", "PorePressure", "Speed", or "Gradient".

For an unconfined solution, "PorePressure" extends above the water table into the unsaturated zone, contouring negative pore pressure (suction) with a tension cut-off at "SuctionCap" metres of head (default 1 m ≈ 9.81 kPa). Set "IncludeSuction" -> False for the saturated zone only. Other options: "PlotPoints" (60) and any ContourPlot option.

PorePressureField

PorePressureField[sol] — returns a pure function u[x,y] giving pore pressure in kPa: the FEM value below the water table and capped hydrostatic suction above it (tension cut-off at "SuctionCap" metres, default 1 m), and Missing["OutsideSoil"] outside the soil body. Sample it on a grid to build a pore-pressure field for export to a slope-stability program:

```
u = PorePressureField[sol, "SuctionCap" -> 1.];
grid = Flatten[Table[Module[{v = u[x, y]},
     If[MissingQ[v], Nothing, {x, y, Round[v, 0.1]}]],
    {x, 0., 75., 2.}, {y, -2., 10., 0.5}], 1];
Export["pwp_grid.csv", Prepend[grid, {"x_m", "y_m", "u_kPa"}]];
Export["phreatic.csv",
  Prepend[sol["PhreaticSurface"], {"x_m", "z_watertable_m"}]];
```

Divide u_kPa by 9.81 for pressure head in metres.

BoundaryFlux, ExitGradient

BoundaryFlux[sol, pred] integrates the outward Darcy flux over the boundary selected by predicate pred (positive = outflow). ExitGradient[sol, pred] returns the maximum hydraulic-gradient magnitude on that boundary — the piping / heave check; compare with the critical gradient (≈ 1 for typical soils).

TerraPercolatio, TerraPercolatioClear — named workspaces

Pass "Name" -> "id" to any solver to store its result:

```
SeepageUnconfined[spec, "Name" -> "damA"];
TerraPercolatio["damA"]      (* retrieve the stored solution *)
TerraPercolatio[]            (* list stored names *)
TerraPercolatio["damA", sol] (* store an association manually *)
TerraPercolatioClear[]       (* forget all; or ["damA"] for one *)
```

Several independent problems can then coexist in one notebook; re-running one name overwrites only that entry and never disturbs the others.

---

6. Solution association keys

Every solver returns an `Association`. Common keys:

Key	Meaning
`"Head"`	interpolating function h[x,y] (total head, m)
`"Psi"`	stream function ψ[x,y]
`"Q"`	discharge (m^3/s per metre run; m^3/s if axisymmetric)
`"Velocity"`	{vx, vy}[x,y] (m/s)
`"Gradient"`	{∂h/∂x, ∂h/∂y}[x,y]
`"PorePressure"`	9810 (h - y) [x,y] (Pa)
`"Mesh"`, `"Layers"`, `"HeadRange"`	mesh, layer list, {min,max} head

Unconfined solutions add `"PhreaticSurface"`, `"ExitPoint"`, `"SaturatedRegion"`, `"GroundSurface"`, `"UnsaturatedZone"`, `"UnsaturatedThickness"`, `"WaterTableDaylights"`, `"Converged"`, `"Iterations"`. Wells add `"DupuitQ"`.

---

7. Worked examples

These are the worked examples from the demonstration notebook (`TerraPercolatioDemo.nb`), reproduced here with their code. Evaluate the "Load the package" cell first, then each example in order.

```
If[FailureQ[Quiet[Needs["TerraPercolatio`"]]],
  Get[FileNameJoin[{NotebookDirectory[], "TerraPercolatio.wl"}]]];
Names["TerraPercolatio`*"]
```

Example 1 — Flow into a dewatered excavation behind sheet piles

A sheet-pile wall (modelled as a thin slit) retains a 3 m deep excavation in a 10 m permeable stratum over an impermeable base. Upstream there are 2 m of standing water above ground level; downstream the ground is excavated to 7 m and dewatered, so the water level in the excavation coincides with the excavated ground level.

```
gUp = 10.;    (* upstream ground level, m *)
gExc = 7.;    (* excavated ground level = water level in the excavation, m *)
hUp = 12.;    (* upstream water level: 2 m standing water above ground, m *)
hDn = gExc;   (* downstream head: excavation dewatered to floor level, m *)
ytip = 4.;    (* pile tip elevation: 6 m below u/s ground, 3 m below floor, m *)
w = 0.05;     (* half-thickness of the pile slit, m *)
L = 20.;      (* lateral extent of the model either side of the wall, m *)
kSand = 1.*^-5;  (* isotropic permeability, m/s *)

excPoly = {{-L, 0.}, {L, 0.}, {L, gExc}, {w, gExc}, {w, ytip}, {-w, ytip}, {-w, gUp}, {-L, gUp}};
excLayers = {<|"Polygon" -> excPoly, "kx" -> kSand, "ky" -> kSand|>};
excHeadBCs = {{hUp, Function[{x, y}, y >= gUp - 10.^-6 && x <= -w]},   (* u/s ground surface *)
```

```
   {hDn, Function[{x, y}, y >= gExc - 10.^-6 && x >= w]};             (* excavation floor *)
pilePred  = Function[{x, y}, Abs[x] <= w + 10.^-6 && y >= ytip - 10.^-6]; (* wall = streamline *)
outerPred = Function[{x, y}, y <= 10.^-6 || Abs[x] >= L - 10.^-6];        (* base & far field *)

excSol = SeepageSolve[excLayers, excHeadBCs, {pilePred, outerPred}, "MeshSize" -> 0.4];
excSol["Q"]
```

The `"WaterLevels"` option draws free-water surfaces (with the water-table symbol), including standing water above ground:

```
FlowNet[excSol, "PotentialDrops" -> 10,
  "WaterLevels" -> {{hUp, {-L, -w}}, {hDn, {w, L}}},
  PlotRange -> {{-L - 0.5, L + 0.5}, {-0.5, 12.8}}, ImageSize -> 900]
```

Exit gradient across the excavation floor next to the wall (piping/heave check — compare with the critical gradient, about 1):

```
ExitGradient[excSol, Function[{x, y}, y >= gExc - 10.^-3 && w + 0.01 <= x <= 3.]]
SeepagePlot[excSol, "PorePressure", ImageSize -> 600]
```

Example 1b — Same excavation, two-layer anisotropic soil

The stratum below 5 m is finer and anisotropic (kx = 2e-6, ky = 5e-7 m/s). Adjacent layer polygons must share vertices along interfaces; a layer interrupted by the wall is supplied as two polygons with the same k. Note the streamline refraction at the interface.

```
yInt = 5.;   (* layer interface elevation, m *)
lowerP  = {{-L, 0.}, {L, 0.}, {L, yInt}, {w, yInt}, {w, ytip}, {-w, ytip}, {-w, yInt}, {-L, yInt}};
upLeft  = {{-L, yInt}, {-w, yInt}, {-w, gUp}, {-L, gUp}};
upRight = {{w, yInt}, {L, yInt}, {L, gExc}, {w, gExc}};
excLayers2 = {<|"Polygon" -> upLeft, "kx" -> 1.*^-5, "ky" -> 1.*^-5|>,
   <|"Polygon" -> upRight, "kx" -> 1.*^-5, "ky" -> 1.*^-5|>,
   <|"Polygon" -> lowerP, "kx" -> 2.*^-6, "ky" -> 5.*^-7|>};
excSol2 = SeepageSolve[excLayers2, excHeadBCs, {pilePred, outerPred}, "MeshSize" -> 0.4];
FlowNet[excSol2, "PotentialDrops" -> 10, "FlowChannels" -> 6, ImageSize -> 900]
```

Example 2 — Unconfined flow through a homogeneous earth dam

A 12 m high dam with 1V:2H slopes and a 4 m crest retains 10 m of water. The phreatic surface (dashed) and the exit point on the downstream slope are found iteratively; the seepage face below the exit point gets the h = y condition automatically.

```
damSpec = <|"Profile" -> {{0., 0.}, {52., 0.}, {28., 12.}, {24., 12.}},
   "UpstreamLevel" -> 10., "DownstreamLevel" -> 0.,
   "kx" -> 1.*^-6, "ky" -> 1.*^-6|>;
damSol = SeepageUnconfined[damSpec, "Points" -> 20];
{damSol["Q"], damSol["ExitPoint"], damSol["Iterations"], damSol["Converged"]}
FlowNet[damSol, "PotentialDrops" -> 12, "FlowChannels" -> 4, ImageSize -> 850]
```

Example 2b — Zoned dam with a clay core

Same dam with a central low-permeability core (k = 5e-8 vs 1e-5 m/s shells). Layers are listed in priority order — the first polygon containing a point supplies its conductivity. Nearly all head is dissipated across the core and the phreatic surface plunges to a low level in the free-draining downstream shell.

```
coreSpec = <|"Profile" -> {{0., 0.}, {52., 0.}, {28., 12.}, {24., 12.}},
   "UpstreamLevel" -> 10., "DownstreamLevel" -> 0.,
   "Layers" -> {
     <|"Polygon" -> {{20., 0.}, {32., 0.}, {28., 12.}, {24., 12.}}, "kx" -> 5.*^-8, "ky" -> 5.*^-8|>,
     <|"Polygon" -> {{0., 0.}, {52., 0.}, {28., 12.}, {24., 12.}}, "kx" -> 1.*^-5, "ky" -> 1.*^-5|>}|>;
```

```
coreSol = SeepageUnconfined[coreSpec, "Points" -> 24];
{coreSol["Q"], coreSol["ExitPoint"]}
FlowNet[coreSol, "PotentialDrops" -> 12, "FlowChannels" -> 3, ImageSize -> 850]
```

Example 3 — TerraLapsus slope: seepage through a 1:2 slope

A 10 m high 1V:2H slope. The upstream water table stands 5 m above toe ground level; downstream the water is at ground level. The impermeable base is 2 m below the base of the slope. The datum (y = 0) is the toe ground level. The solver finds the phreatic surface, the seepage-face exit point, and Q.

```
Hslope = 10.;   (* slope height, m *)
mSlope = 2.;    (* slope 1V : mH, i.e. 1:2 *)
yToe = 0.;      (* toe ground level = datum, m *)
yTop = yToe + Hslope;   (* upper ground level, m *)
yBase = yToe - 2.;      (* impermeable base, 2 m below base of slope, m *)
huSlope = 5.;   (* u/s water table, 5 m above toe level, m *)
hdSlope = 0.;   (* d/s water at ground level, m *)
kSlope = 1.*^-5;  (* isotropic permeability, m/s *)
xCrest = 30.;   (* crest position, m *)
xToe = xCrest + mSlope Hslope;   (* toe at x = 50 m *)
xEnd = 80.;     (* 30 m of lower ground downstream, m *)

slopeProfile = {{0., yBase}, {xEnd, yBase}, {xEnd, yToe}, {xToe, yToe}, {xCrest, yTop}, {0., yTop}};
slopeSpec = <|"Profile" -> slopeProfile,
   "UpstreamLevel" -> huSlope, "DownstreamLevel" -> hdSlope,
   "UpstreamFace" -> {{0., yBase}, {0., yTop}},
   "DownstreamFace" -> {{xEnd, yBase}, {xEnd, yToe}, {xToe, yToe}, {xCrest, yTop}},
   "kx" -> kSlope, "ky" -> kSlope|>;

slopeSol = SeepageUnconfined[slopeSpec, "Points" -> 24];
{slopeSol["Q"], slopeSol["ExitPoint"], slopeSol["Iterations"], slopeSol["Converged"]}
FlowNet[slopeSol, "PotentialDrops" -> 10, "FlowChannels" -> 4, ImageSize -> 950]
```

The phreatic surface and pore pressures below it feed a slope-stability analysis:

```
slopeSol["PhreaticSurface"]
slopeSol["PorePressure"][40., 0.]/1000.   (* pore pressure in kPa at a point *)
```

Example 3b — Rainfall on the slope (raised water table, unsaturated zone)

Surface infiltration is added with `"Rainfall"` in mm/day. The rain percolates vertically through the unsaturated zone and recharges the water table beneath, raising it and the pore pressures. The water table is capped at the ground surface; the gap is returned as `"UnsaturatedZone"`. Re-using `slopeSpec`, compare dry ground with 40 mm/day of rain:

```
slopeWet = SeepageUnconfined[slopeSpec, "Points" -> 24, "Rainfall" -> 40.];
{slopeWet["Q"], slopeWet["UnsaturatedThickness"], slopeWet["WaterTableDaylights"],
 slopeWet["PorePressure"][40., 0.]/1000. - slopeSol["PorePressure"][40., 0.]/1000.}
FlowNet[slopeWet, "PotentialDrops" -> 10, "FlowChannels" -> 4, ImageSize -> 950]
```

`"UnsaturatedThickness"` is {min, max} of the gap between water table and ground; when the minimum falls to zero the water table daylights on the slope face and that stretch becomes a runoff/seepage face. Increase the rainfall until it daylights to find the critical intensity.

Example 3c — Named problems and a pore-pressure map with suction

Pass `"Name" -> "id"` to any solver to store its result under `TerraPercolatio["id"]`, so independent problems coexist without clashes:

```
SeepageUnconfined[slopeSpec, "Name" -> "dry", "Points" -> 24];
SeepageUnconfined[slopeSpec, "Name" -> "wet", "Points" -> 24, "Rainfall" -> 40.];
{TerraPercolatio[],
 TerraPercolatio["dry"]["Rainfall_mmday"], TerraPercolatio["wet"]["Rainfall_mmday"]}
```

The pore-pressure map extends above the water table into the unsaturated zone, contouring suction clipped to `"SuctionCap"` metres of head (default 1 m ~ 9.81 kPa) — the tension cut-off used in slope stability:

```
SeepagePlot[TerraPercolatio["wet"], "PorePressure", "SuctionCap" -> 1., ImageSize -> 820]
```

Example 3d — Anisotropic slope with a thin permeable drainage seam

A strongly anisotropic slope soil (kx = 1e-7 m/s, ky = 1e-9 = 0.01 kx, so flow is almost horizontal) contains a thin, highly permeable seam from y = 4 to y = 4.5 (kx = ky = 1e-5, 100x the bulk kx) acting as an internal drainage layer. Layers are searched in order, so the seam polygon (listed first) takes priority in its band. A small `"MeshSize"` resolves the 0.5 m seam.

```
baseProfile = {{0., -2.}, {75., -2.}, {75., 0.}, {50., 0.}, {30., 10.}, {0., 10.}};
seam = <|"Polygon" -> {{-5., 4.}, {80., 4.}, {80., 4.5}, {-5., 4.5}},
   "kx" -> 1.*^-5, "ky" -> 1.*^-5|>;              (* permeable drainage seam *)
baseSoil = <|"Polygon" -> baseProfile, "kx" -> 1.*^-7, "ky" -> 1.*^-9|>;  (* ky = 0.01 kx *)
seamSpec = <|"Profile" -> baseProfile, "UpstreamLevel" -> 8., "DownstreamLevel" -> 0.,
   "UpstreamFace" -> {{0., -2.}, {0., 10.}},
   "DownstreamFace" -> {{75., -2.}, {75., 0.}, {50., 0.}, {30., 10.}},
   "Layers" -> {seam, baseSoil}|>;                  (* seam first = higher priority *)

seamSol = SeepageUnconfined[seamSpec, "Points" -> 26, "MeshSize" -> 0.15, "Name" -> "seamSlope"];
{seamSol["Q"], seamSol["ExitPoint"], seamSol["Converged"]}
FlowNet[seamSol, "PotentialDrops" -> 10, "FlowChannels" -> 5, ImageSize -> 900]
```

Streamlines crowd into the seam and equipotentials refract sharply across it — most discharge is carried horizontally along the seam to the toe. The head map shows the head staying high above the seam (perched) while the seam relieves the slope below:

```
SeepagePlot[seamSol, "Head", "IncludeSuction" -> False, Contours -> 20, ImageSize -> 900]
```

Example 3e — Slope under rainfall: pore pressures for stability (TerraLapsus)

The anisotropic slope with the drainage seam, run under rainfall, producing a pore-pressure field for a limit-equilibrium program. `PorePressureField[sol]` returns u[x,y] in kPa (FEM below the water table, capped suction above); sampling it on a grid gives the field to import.

Physics note: rainfall can only infiltrate as fast as the soil's vertical permeability ky allows. If the applied rate exceeds ky the water table mounds unrealistically (the surplus should run off), so the base soil here uses ky = 0.1 kx (1e-7 m/s) with rainfall 5 mm/day (~5.8e-8 m/s < ky). With the near-impermeable ky of Example 3d, rain would simply run off.

```
baseProfile = {{0., -2.}, {75., -2.}, {75., 0.}, {50., 0.}, {30., 10.}, {0., 10.}};
seam = <|"Polygon" -> {{-5., 4.}, {80., 4.}, {80., 4.5}, {-5., 4.5}},
   "kx" -> 1.*^-5, "ky" -> 1.*^-5|>;
baseSoil = <|"Polygon" -> baseProfile, "kx" -> 1.*^-6, "ky" -> 1.*^-7|>;  (* ky = 0.1 kx *)
slopeRainSol = SeepageUnconfined[<|"Profile" -> baseProfile,
    "UpstreamLevel" -> 8., "DownstreamLevel" -> 0.,
    "UpstreamFace" -> {{0., -2.}, {0., 10.}},
    "DownstreamFace" -> {{75., -2.}, {75., 0.}, {50., 0.}, {30., 10.}},
    "Layers" -> {seam, baseSoil}|>,
   "Points" -> 24, "MeshSize" -> 0.15, "Rainfall" -> 5., "Name" -> "slopeRain"];
{slopeRainSol["Q"], slopeRainSol["HeadRange"], slopeRainSol["WaterTableDaylights"]}

SeepagePlot[slopeRainSol, "PorePressure", "SuctionCap" -> 1., ImageSize -> 860]
```

```
u = PorePressureField[slopeRainSol, "SuctionCap" -> 1.];
grid = Flatten[Table[Module[{v = u[x, y]},
     If[MissingQ[v], Nothing, {x, y, Round[v, 0.1]}]],
    {x, 0., 75., 2.}, {y, -2., 10., 0.5}], 1];
Export["slope_pwp_grid.csv", Prepend[grid, {"x_m", "y_m", "u_kPa"}]];
Export["slope_phreatic.csv",
  Prepend[Round[slopeRainSol["PhreaticSurface"], 0.01], {"x_m", "z_watertable_m"}]];
{Length[grid], MinMax[grid[[All, 3]]]}
```

Example 4 — Axisymmetric flow to a pumped well (drawdown cone)

Unconfined radial flow to a fully penetrating well on an impermeable base, solved in (r, z). `SeepageWell` finds the phreatic cone of depression and the seepage face on the screen. Compare Q with the Dupuit-Thiem formula, which ignores the seepage face.

```
rWell = 0.3;    (* well radius, m *)
Rinf = 100.;    (* radius of influence, m *)
Haq = 20.;      (* far-field water table above the base, m *)
hWell = 8.;     (* pumped water level in the well, m *)
krWell = 1.*^-4;  kzWell = 1.*^-4;   (* radial / vertical permeability, m/s *)

wellSpec = <|"WellRadius" -> rWell, "OuterRadius" -> Rinf,
   "FarWaterTable" -> Haq, "WellWaterLevel" -> hWell, "kr" -> krWell, "kz" -> kzWell|>;
wellSol = SeepageWell[wellSpec, "Points" -> 30];
{wellSol["Q"], wellSol["DupuitQ"], wellSol["ExitPoint"], wellSol["Converged"]}
FlowNet[wellSol, "PotentialDrops" -> 12, "FlowChannels" -> 8,
  "WaterLevels" -> {{hWell, {0., rWell}}}, PlotRange -> {{0, 60}, {0, 22}}, ImageSize -> 900]
```

Example 5 — 3D flow into a rectangular excavation

A 10 m x 16 m excavation behind sheet-pile walls, dewatered to a floor 3 m below ground, in a 10 m stratum. One quarter is modelled with symmetry planes at x = 0 and y = 0. Walls and the excavated void are subtracted by constructive solid geometry. (No 2D stream function exists in 3D, so section StreamPlots and flux integrals replace flow nets.)

```
D3 = 10.;    (* stratum thickness: base z = 0, ground z = 10 *)
axq = 5.;    (* excavation half-length in x (full 10 m) *)
byq = 8.;    (* excavation half-width in y (full 16 m) *)
tw = 0.4;    (* wall thickness, m *)
ztip3 = 4.;  (* wall toe elevation: 6 m walls, 3 m embedment *)
zf = 7.;     (* excavation floor = water level inside (dewatered), m *)
hFar = 10.;  (* far-field head: water table at ground level, m *)
LxQ = 40.; LyQ = 40.;   (* plan extent of the quarter model, m *)
k3 = 1.*^-5; (* isotropic permeability, m/s *)

wallX = Cuboid[{axq, 0., ztip3}, {axq + tw, byq + tw, D3}];
wallY = Cuboid[{0., byq, ztip3}, {axq + tw, byq + tw, D3}];
void  = Cuboid[{0., 0., zf}, {axq, byq, D3}];
reg3  = RegionDifference[Cuboid[{0., 0., 0.}, {LxQ, LyQ, D3}], RegionUnion[wallX, wallY, void]];
mesh3 = ToElementMesh[reg3, MaxCellMeasure -> 30.,
   MeshRefinementFunction -> Function[{v, vol},
     Module[{c = Mean[v]}, vol > If[c[[1]] < 10 && c[[2]] < 13, 0.5, 30.]]], "MeshOrder" -> 2];

h3 = NDSolveValue[{Laplacian[h[x, y, z], {x, y, z}] == 0,
    DirichletCondition[h[x, y, z] == hFar, x >= LxQ - 10.^-4 || y >= LyQ - 10.^-4],
    DirichletCondition[h[x, y, z] == zf,
```

```
     z >= zf - 10.^-4 && z <= zf + 10.^-4 && x <= axq + 10.^-4 && y <= byq + 10.^-4]},
   h, {x, y, z} \[Element] mesh3];

(* total inflow and exit gradients; the pit corner is more critical than mid-wall *)
dzh = Derivative[0, 0, 1][h3]; dxh = Derivative[1, 0, 0][h3];
Qtot3 = 4. NIntegrate[k3 dzh[x, y, zf - 10.^-3], {x, 0., axq}, {y, 0., byq},
    AccuracyGoal -> 4, PrecisionGoal -> 4];
{Abs[Qtot3],
 Abs[dzh[axq - 0.3, 1., zf - 10.^-3]],          (* mid long wall *)
 Abs[dzh[axq - 0.3, byq - 0.3, zf - 10.^-3]],   (* corner  *)
 Abs[dzh[0.1, 0.1, zf - 10.^-3]]}               (* centre  *)
```

A section on the symmetry plane y = 0 (equipotentials + streamlines) and a plan of the drawdown at mid-depth are drawn with `ContourPlot`/`StreamPlot`; see the demonstration notebook for the full plotting code.

Example 6 — Parallel drainage trenches at a spacing (water-table lowering)

Design of linear drains to lower the water table for slope stability: trenches at spacing S collect steady infiltration q. One half-cell is modelled from a drain to the mid-spacing divide (`"UpstreamNoFlow" -> True`). Recharge crosses the phreatic surface (`"Recharge" -> q`). The design check is the water-table apex midway between drains, which the Donnan/Hooghoudt formula approximates.

```
Sdr = 20.;    (* drain spacing (centre to centre), m *)
wdr = 0.5;    (* drain trench half-width, m *)
hDrain = 1.5; (* water level in the drain above the base, m *)
zGnd = 6.;    (* ground level above the base, m *)
qInf = 5.*^-7;(* steady infiltration, m/s (~43 mm/day) *)
kDr = 1.*^-5; (* soil permeability, m/s *)
Ldr = Sdr/2 - wdr;   (* flow length: divide to drain wall = 9.5 m *)

drainSpec = <|"Profile" -> {{0., 0.}, {Ldr, 0.}, {Ldr, zGnd}, {0., zGnd}},
   "UpstreamLevel" -> 3.,       (* initial guess for the apex *)
   "DownstreamLevel" -> hDrain,
   "UpstreamFace" -> {{0., 0.}, {0., zGnd}},      (* mid-spacing divide *)
   "DownstreamFace" -> {{Ldr, 0.}, {Ldr, zGnd}},  (* drain wall *)
   "kx" -> kDr, "ky" -> kDr|>;

drainSol = SeepageUnconfined[drainSpec, "Points" -> 24, "Recharge" -> qInf, "UpstreamNoFlow" -> True];
{drainSol["Q"], qInf Ldr,                        (* outflow vs recharge balance *)
 drainSol["PhreaticSurface"][[1, 2]],            (* FEM water-table apex *)
 Sqrt[hDrain^2 + qInf Ldr^2/kDr]}                (* Donnan estimate *)
FlowNet[drainSol, "PotentialDrops" -> 8, "FlowChannels" -> 8,
  "WaterLevels" -> {{hDrain, {Ldr, Ldr + 2 wdr}}},
  PlotRange -> {{-0.2, Ldr + 2 wdr + 0.2}, {-0.3, zGnd + 0.4}}, ImageSize -> 800]
```

Change `Sdr` and re-run; the apex height is the number to check against the required water-table level.

Example 7 — Line of wellpoints at a spacing (excavation dewatering)

Design of a line of wellpoints: one well per spacing s is modelled in a 3D slab between symmetry planes (through the well and midway to the next). The well-line plane is no-flow, so supply comes from one side. The design outputs are the discharge per well and the residual head midway between wells.

```
sWp = 3.;     (* wellpoint spacing, m *)
rWp = 0.25;   (* effective wellpoint radius, m *)
DWp = 10.;    (* saturated thickness: base z = 0, water table z = 10 *)
HWp = 10.;    (* far-field head, m *)
hwWp = 3.;    (* operating (suction) level in the wellpoints, m *)
```

```
zScr = 8.;    (* top of screen: screened from the base to 8 m *)
LWp = 50.;    (* distance to the far-field supply boundary, m *)
kWp = 1.*^-5; (* permeability, m/s *)

regW = RegionDifference[Cuboid[{0., 0., 0.}, {LWp, sWp/2, DWp}],
   Cylinder[{{0., 0., -0.5}, {0., 0., zScr}}, rWp]];
meshW = ToElementMesh[regW, MaxCellMeasure -> 30.,
   MeshRefinementFunction -> Function[{v, vol},
     Module[{r = Norm[Mean[v][[1 ;; 2]]]}, vol > Which[r < 2., 0.01, r < 8., 0.5, True, 30.]]],
   "MeshOrder" -> 2];
hW = NDSolveValue[{Laplacian[h[x, y, z], {x, y, z}] == 0,
    DirichletCondition[h[x, y, z] == HWp, x >= LWp - 10.^-4],
    DirichletCondition[h[x, y, z] == hwWp, x^2 + y^2 <= (rWp + 0.02)^2 && z <= zScr + 10.^-3]},
   h, {x, y, z} \[Element] meshW];

dxhW = Derivative[1, 0, 0][hW]; dzhW = Derivative[0, 0, 1][hW];
QperWell = 2. Abs[NIntegrate[kWp dxhW[LWp - 10.^-3, y, z],
     {y, 0., sWp/2}, {z, 0., DWp}, AccuracyGoal -> 4, PrecisionGoal -> 4]];
{QperWell,                          (* discharge per wellpoint, m^3/s *)
 hW[0.011, sWp/2, DWp - 10.^-3],    (* residual head midway between wells *)
 hW[0.011, sWp/2, 5.]}              (* ditto at mid-screen depth *)
```

A section through a well and a plan of the water table near the line are drawn with `ContourPlot`/`StreamPlot`; see the demonstration notebook for the full plotting code.

---

8. Conventions and modelling tips

- Geometry in metres, k in m/s, heads in metres of water.
- Boundary predicates are `(x,y) -> True/False`; give them tolerances (`y <= 0.001`) rather than exact equality.
- Adjacent layer polygons must share interface vertices; a layer cut by a wall is two polygons with the same k.
- Model sheet piles / cutoff walls as thin slits (width $\approx$ L/400).
- `"MeshSize"` is the maximum element area — halve it to check convergence of Q, and refine it to resolve thin layers or seams in unconfined multi-layer runs (which use pointwise conductivities on a single mesh).
- Keep rainfall below the soil's vertical conductivity ky, or expect the water table to mound to the ground surface (the surplus is runoff).

---

9. Validation

- Two-layer 1-D series flow matches the analytic interface head to machine precision.
- Sheet-pile discharge matches classical flow-net solutions ($Q/k\Delta H \approx 0.48$ for the demo excavation).
- Earth-dam exit point and Q agree with the Schaffernak/Casagrande approximations; the slope Q agrees with the Dupuit estimate to ≈ 5 %.
- The pumped-well Q matches Dupuit–Thiem within 0.5 % while additionally resolving the seepage face on the screen.
- The parallel-drain water-table apex matches the Donnan/Hooghoudt formula within 2 % and balances recharge against drain outflow.

---

10. Limitations

Steady state only; no consolidation or transient storage. The unsaturated zone is represented by capped hydrostatic suction, not a Richards-equation solution. Unconfined multi-layer problems use pointwise conductivities on a

non-conforming mesh (refine near thin layers). Free surfaces are assumed single-valued in x. There is no 3D stream function, so 3D problems use section StreamPlots and flux integrals rather than flow nets.

---

11. References

- Casagrande, A. (1937). *Seepage through dams.* J. New England Water Works Assoc.
- Cedergren, H. R. (1989). *Seepage, Drainage, and Flow Nets*, 3rd ed. Wiley.
- Harr, M. E. (1962). *Groundwater and Seepage.* McGraw-Hill.
- Hooghoudt, S. B. (1940); Donnan, W. W. (1946) — drain-spacing theory.

---

12. License

Released under the MIT License, declared in the paclet metadata (`"License" -> "MIT"` in `PacletInfo.wl`).