

Limitas

Slope stability, quick start

TerraLapsus Part 1 · version V1

August 27, 2026

Describe the ground once. Then choose a method. Every method consumes the same model through the same slice builder, so a difference between two answers is a difference in method and not in the problem posed.

This is the short manual. It covers running the program. It does not derive the equations or justify the choices; the full manual does that.

Contents

1	Load it	1
2	Describe the ground	2
2.1	Geometry — two forms	2
2.2	Materials	2
2.3	Water	2
2.4	Tension crack	3
2.5	Partial factors	4
2.6	Look at it	4
3	Run a method	5
3.1	The fast path	5
3.2	Letting each method search for itself	5
3.3	Which method	5
3.4	Grid cost	6
3.5	Everything below a threshold	6
4	Read the answer	6
5	Seepage	9
6	When a circle is the wrong shape	10
7	Traps	11
8	What is in the folder	12

1 Load it

```
Get["/path/to/TerraLapsus Part 1 V1/Limitas.wl"];
LimitasHelp[]
```

One line. It loads five stability methods, two non-circular searches and the seepage solver, and puts them all on `$ContextPath` so you call them by plain name. It also reads the two add-ons that sit beside it — `Criticality.wl`, section 3.5, and `ThrustLine.wl`, section 4 — so there is nothing else to `Get` and no way to load them in the wrong order. If either file is missing the loader says so once and carries on without it.

```
LimitasParallelSetup[];      (* once per session; about 8x on the searches *)
```

After any `Get`, rebuild your models. A model built before a reload is stale.

2 Describe the ground

2.1 Geometry — two forms

```
(* parametric planar slope *)
geom = <|"Height" -> 10., "SlopeAngle" -> 26.57 Degree,
      "FoundationDepth" -> 10., "CrestLength" -> 22.,
      "ToeLength" -> 22., "XStart" -> -12.|>;

(* or an arbitrary profile, crest to toe *)
geom = <|"Points" -> {{-40., 20.}, {-20., 20.}, {-0.5, 10.}, {30., 10.}}|>;
```

Nothing assumes the model starts at $x = 0$. x increases from crest towards toe, so the ground descends with increasing x ; y is elevation above a datum treated as rigid base; sliding is left to right.

30 is 30 radians. Write `30. Degree`.

2.2 Materials

Regions are listed *top down*. "Base" is the elevation of the bottom of that region — a number, a polyline, or `Automatic` for the lowest one. A region whose base rises above the one over it simply pinches out.

"Base" on the DEEPEST region is ignored. That region always extends down to the rigid base, so that no void can open between the bottom of the soil and the level below which trial circles are refused. On a single-region model this means your "Base" never does anything at all. To move the bottom of the model, set "BaseElevation":

```
model = SlopeModel[geom, mat, "BaseElevation" -> -5.];
```

which moves the soil, the floor of the search window and the rejection depth together. On the `QuickStart` slope that lets the search reach $y = -4.6$ instead of 0.4 and gives $F = 1.40071$ against 1.40646 . "Base" on any region *above* the deepest is honoured as written, and a mismatch on the deepest one now raises `SlopeModel::deepbase` rather than being discarded in silence.

```
mat = <|"Regions" -> {
  <|"Name" -> "Fill", "Base" -> 6., "UnitWeight" -> 20.,
    "Cohesion" -> 10., "FrictionAngle" -> 25. Degree,
    "Permeability" -> 1.*^-6|>,
  <|"Name" -> "Clay", "Base" -> Automatic, "UnitWeight" -> 20.,
    "Cohesion" -> 15., "FrictionAngle" -> 20. Degree,
    "Permeability" -> {2.*^-8, 5.*^-9}|>|>;
```

2.3 Water

Three ways, in increasing order of honesty:

"Cohesion"	c' in kPa
"FrictionAngle"	ϕ' , in radians unless you write Degree
"UnitWeight"	γ in kN/m ³
"SaturatedUnitWeight"	used below the water table if given
"Permeability"	k in m/s; a number is isotropic, {kx, ky} anisotropic. Only needed for seepage

```

"PorePressure" -> <|"WaterTable" -> 13.|>      (* flat table *)
"PorePressure" -> <|"WaterTable" -> {{x,y}, ...}|> (* piecewise linear *)
"PorePressure" -> <|"Ru" -> 0.3|>                (* pore pressure ratio *)
"PorePressure" -> SeepagePoreSpec[sol]           (* computed - section 5 *)

```

2.4 Tension crack

```

"TensionCrack" -> <|"Depth" -> Automatic,      (* or a depth, or "Critical" *)
                  "WaterDepth" -> Full|>      (* or 0, or a depth *)

```

A vertical crack at the head of the mechanism, running from the ground surface down to where its base meets the slip surface. It does three things, and only the first is obvious:

1. it removes the wedge behind it from the driving weight;
2. it removes that length of slip surface, and with it the shear resistance the removed arc was providing;
3. if it holds water it applies a horizontal thrust $P = \gamma_w z_w^2 / 2$ at $z_w / 3$ above the crack base, which drives.

Leave out the second and a crack looks unconditionally stabilising. It is not. On the QuickStart slope a *dry* crack lowers the factor of safety, because the arc it removes was resisting more than the wedge it removes was driving:

	Bishop	Janbu	crack depth
no crack	1.40646	1.41850	—
dry, Automatic depth	1.40318	1.40632	1.428 m
water-filled, Automatic	1.38865	1.38554	1.428 m
water-filled, "Critical"	1.36828	1.34259	3.5 m

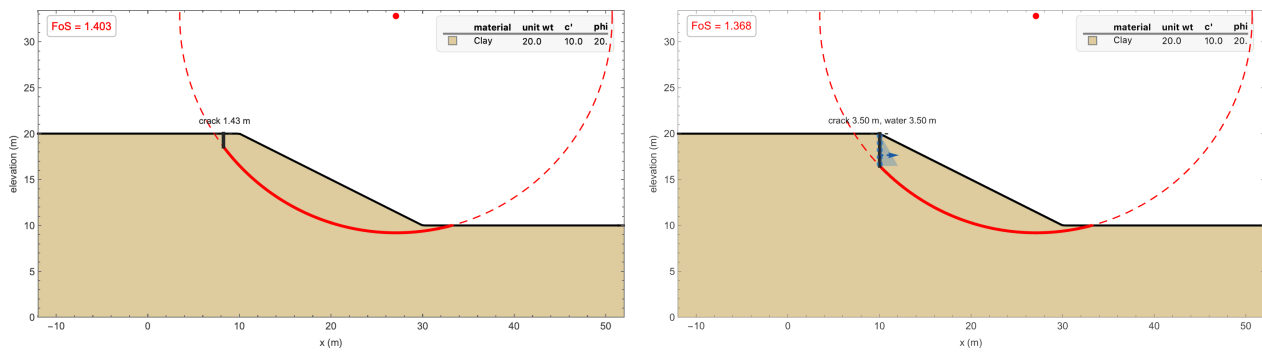


Figure 1: The crack is drawn from the slice array, not re-derived, so the picture cannot disagree with the analysis. Left: dry, **Automatic** depth. Right: 3.5 m and water-filled — the blue column is the water, the shaded triangle the hydrostatic pressure on the face, widest at the base, and the arrow its resultant at $z_w/3$ above the crack base, which is exactly the force the equations use. **The red arc stops at the crack** in both: drawing the full circle would show a slip surface running up through ground the analysis has already removed.

Automatic is the Rankine active depth $z_c = 2c'/(\gamma\sqrt{K_a})$ at the factored strengths. **It is not the worst depth**, and the last row is why: with water in it the worst depth is far deeper, because the thrust grows as z_w^2 while the resisting arc it costs you grows more slowly. "**Critical**" scans for the depth giving the lowest F — coarsely, then refined — and on this slope finds 3.5 m, two and a half times Rankine and 1.5 % lower again on Bishop.

"Critical" multiplies the cost of every evaluation by the number of depths tried, eighteen by default. Resolve it once on the critical circle, then pin the answer with a number for anything repeated.

Which methods understand it

The truncation lives in the slice builder, so *every* method gets it, and both slice builders apply it — circles and polylines alike. The water thrust is applied by Bishop, Janbu, Morgenstern–Price and Spencer. On the QuickStart circle with a 3.5 m crack filled to the top:

	Bishop	Janbu	MP	Spencer	Sarma
no crack	1.40646	1.41850	1.40415	1.40414	1.40406
crack, dry	1.47361	1.48113	1.47239	1.47191	1.47190
crack, full	1.36828	1.34259	1.36676	1.36560	1.47190

Sarma does not, and the reason is not the crack. Its formulation here carries no external *horizontal* force at all: the vertical component of a surface load reaches it through W , but the horizontal component never enters equations (5.49) to (5.52). Sarma's number in the last row is its dry-crack value, unchanged to every digit. It raises `SliceArray::crackthrust` when asked, and the message says so. See trap 11.

2.5 Partial factors

```
model = SlopeModel[geom, mat, "StrengthFactor" -> 1.];
```

The default is 1/1.25, not 1. That is right for design and wrong for checking against a published number, because it returns an over-design factor on strength that has already been reduced. Set it explicitly and know which you are asking for.

2.6 Look at it

```
SlopeModelPlot[model]
```

Do this before solving anything. Most wrong answers are wrong models, and they are obvious on sight.

3 Run a method

3.1 The fast path

Find the critical circle with Bishop, then evaluate the rigorous methods on that same circle. Every difference is then a difference in the equilibrium conditions satisfied, and nothing else.

```
resB = BishopCriticalCircle[model, "Grid" -> 12, "Parallel" -> True];
resJ = JanbuCritical[model, "Grid" -> 12, "Parallel" -> True];

resMP = MorgensternPriceFoS[model, resB];
resSp = SpencerFoS[model, resB];
resSa = SarmaFoS[model, resB, "Psi" -> "Sarma"];

resB["FoS"]
```

On the Griffiths & Lane 2:1 clay slope of Examples/01-QuickStart.wl the whole block above takes about two seconds and gives:

method	F	satisfies
Bishop simplified	1.40646	moment
Janbu simplified	1.39203	force
Morgenstern–Price	1.40415	both, $\lambda = 0.422$
Spencer	1.40414	both, $\theta = 18.84^\circ$
Sarma (1973)	1.40406	both, $K_c = 0.167$ g

The three full-equilibrium methods inside 0.01%, Bishop just above, Janbu just below. **That pattern is what a healthy run looks like.** Published values for this slope are 1.38 (Bishop) and 1.40 (finite element).

3.2 Letting each method search for itself

MorgensternPriceCritical, SpencerCritical and SarmaCritical exist and take the same arguments. They solve for λ inside every trial circle, so they are far slower — minutes, not seconds. **Reach for them when the mechanism is not circle-like**, not for a first look.

3.3 Which method

Bishop	circles only, moment only. The default first look.
Janbu	any shape, force only. Comes out low; used as the objective by both non-circular searches because it is cheap.
Morgenstern–Price	any shape, full equilibrium, $X = \lambda f(x)E$.
Spencer	the same with $f(x) = 1$, so the interslice resultant holds a constant inclination θ .
Sarma	full equilibrium, and gives the critical horizontal acceleration K_c as well as F . Use it for anything seismic.

3.4 Grid cost

A sweep costs $\text{Grid}^2 \times (\text{Grid} + 1)$ trial circles: `Grid` -> 12 is 1872. Start at 12; raise it, or add "Refinements" -> 2, only for the number you finally quote.

3.5 Everything below a threshold

A search reports one surface and discards the rest of the sweep. That is a pity, because *which ground no acceptable surface avoids* is a different question from *what is the minimum*, and the sweep has already answered both.

```
sub = SubCriticalSurfaces[model, "Threshold" -> 1.5, "Grid" -> 12,
                           "Parallel" -> True];
fld = CriticalityField[model, sub];
CriticalityMapPlot[model, fld]
CriticalitySummary[model, sub]
```

It re-sweeps `CircleSearchGrid`, keeps every circle below the threshold, and rasterises the union of their failure masses. Each point is coloured by the *lowest* factor of safety of any mass containing it, so the dark core is the ground common to the worst mechanisms. The whole thing costs one sweep — 0.9s at "Grid" -> 12 with "Parallel" -> True, 9.4s without — because only the circles that survive the threshold are asked for their geometry.

Ask for 1.3 on this slope and you get nothing. That is correct: the critical circle is 1.4065, so no trial surface qualifies. The label says so and the bare section is drawn.

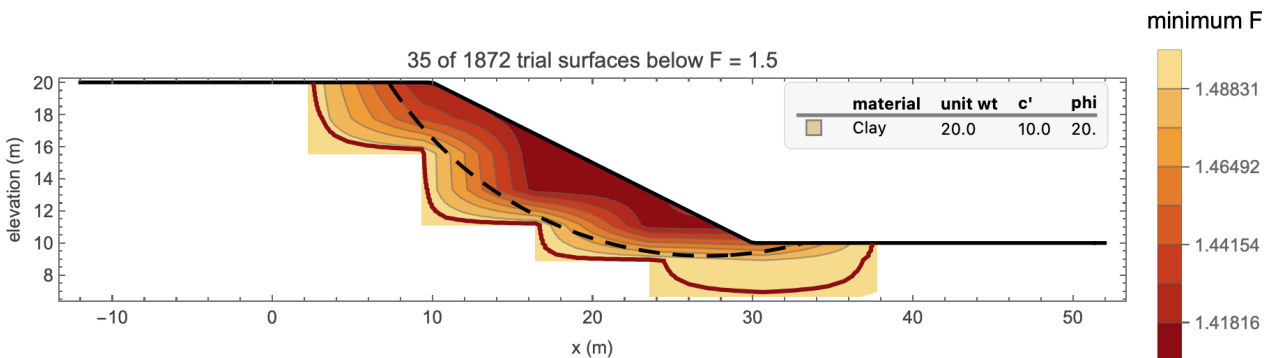


Figure 2: The 2:1 clay slope of `Examples/01-QuickStart.wl`, threshold 1.5. Thirty-five of 1872 trial circles qualify — 2.5% of the 1395 that are kinematically admissible. Dashed, the critical circle at $F = 1.4065$; dark red, the envelope of all sub-critical masses. The stepped outline is real, not a rendering artefact: it is thirty-five discrete circles, and it smooths as "Grid" rises.

"Mode" -> "Count" colours each point by how many sub-critical surfaces contain it instead. **That is a picture of the search, not of the ground.** The trial circles come from a regular grid in centre-and-radius space, so doubling "Grid" roughly octuples every count while the slope stays exactly as it was. Read it to check the search window sits where you think it does; never contour it and call it a probability of failure. The minimum- F field is the one to quote, because a minimum over a denser sample of the same space converges and a count does not.

A zone that touches the edge of the map is a search window that is too small — the same complaint `BishopCriticalCircle::atwindow` makes about a single circle. Check "EnvelopeX" from `CriticalitySummary` against `model["Domain"]` before believing the extent.

4 Read the answer

Every run, without exception. These are *top-level keys of the result*, not a sub-association.

```
KeyTake[resB, {"AtDomainEdge", "AtSearchEdge", "OnRigidBase",
               "UpliftFraction"}]
```

"AtDomainEdge"	the surface daylight at the model boundary. The mass is truncated by the edge, not by the slope, and the number is meaningless. Extend "CrestLength" or "ToeLength"
"AtSearchEdge"	the minimum sits on the search window limit. The true minimum is probably outside it
"OnRigidBase"	bottoming on the deepest allowed surface
"UpliftFraction"	fraction of slices where u exceeds the overburden. Above about 0.15 this is heave or blowout, not sliding , and a sliding factor of safety is answering the wrong question
"WellConditioned"	Janbu and Sarma. False means the answer is not reliable
"ThetaDegrees"	Spencer. Above $\sim 30^\circ$, suspect the surface

Confirm the installation at any time:

```
SarmaCheck[] ["AllPass"]      (* True *)
```

The line of thrust

The factor of safety says nothing about whether the stress state behind it is one the soil could actually carry. The line of thrust does: it is the locus of the points at which the interslice normal forces E act, recovered from the moment equilibrium of each slice.

```
resSp = SpencerFoS[model, resB];
resMP = MorgensternPriceFoS[model, resB];

ThrustLinePlot[model, resSp]
ThrustLineTable[model, resSp]      (* the numbers behind the line *)
ThrustLine[model, resMP] ["InsideFraction"] (* 1. if it never leaves the mass *)
ThrustLine[model, resSp] ["AboveGround"]   (* and if not, which way it left *)
```

Spencer and Morgenstern–Price only. A line of thrust is the answer to the moment equilibrium of a single slice, and that equation has content only where the slice carries an interslice *shear*. Bishop and Janbu simplified set $X = 0$, so their slices cannot satisfy it individually; asking them is asking a question their assumptions have already answered with silence, and the function refuses rather than drawing a meaningless line.

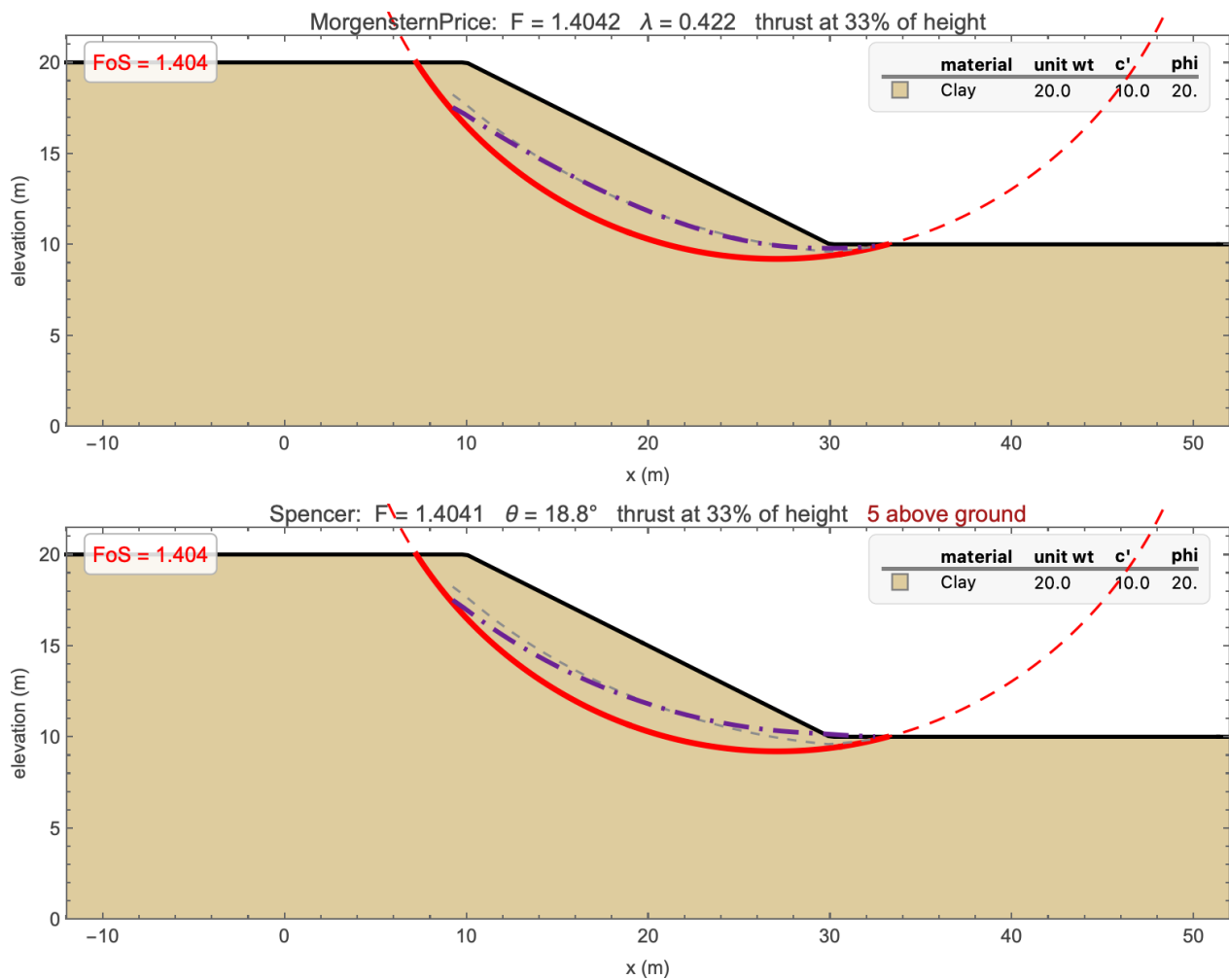


Figure 3: The same critical circle, the same F to four figures, two interslice functions. Purple dot-dash is the line of thrust, grey dashed is one third of the slice height. Through the body of the surface the two are almost indistinguishable and both track the third closely — median height ratio 0.327 for Morgenstern–Price and 0.332 for Spencer. They part company at the *toe*: Morgenstern–Price stays inside the mass throughout, while Spencer rises above the ground surface over the last five boundaries. Neither misbehaves at the crest.

Both ends are dropped rather than drawn. E must return to zero at each end of the surface for horizontal closure, and the thrust elevation is a moment *divided by* E , so the last boundaries are $0/0$ in the limit and numerically wild just before it. "EFloor" sets how much of peak E a boundary needs to be drawn, 2% by default, and "Dropped" counts what that cost.

Read the median, not the mean. The height ratio has a long tail wherever the sliding mass thins to nothing. At the toe of this circle the slice is 0.2 m deep, so a thrust 0.1 m out of place reads as a ratio of 1.2, and five such boundaries pull Spencer's *mean* to 0.447 — a number the line does not go anywhere near. Its median is 0.332. "MeanHeightRatio" and "MedianHeightRatio" are both reported and the plot label uses the median.

What it is for. A third of the height is the classical expectation for a homogeneous mass and the reason it is the usual hand assumption — it is a sanity check, not a criterion the solution must meet. What matters is whether the line stays *inside* the mass, and *where* it does not. E is the resultant of a normal stress distribution on a vertical interface that exists only between the slip surface and the ground; if that stress is compressive throughout, its resultant is a weighted average of elevation with non-negative weights and cannot land outside the two ends. So either excursion means the distribution changes sign — tension somewhere on the interface — and "AboveGround" and "BelowSurface" are counted separately because the tension sits in different places. Above the ground needs tension at the *base* of the interface, with the compression crowded to the top. Below the slip surface is the mirror: tension at the *top*, near the ground, which is the tension-crack condition — and this package models no tension crack, so such a solution leans on tension it has no mechanism to release. Weigh both by where they occur — a boundary just outside in the last 0.2 m of a mass that has thinned to nothing is far weaker evidence than one out in the body of the surface. In the figure above F is 1.404 by both methods and identical to four figures, so the factor cannot tell you which stress state is admissible. Only the thrust line can.

5 Seepage

A water table drawn by hand is a guess, and F is more sensitive to pore pressure than to almost anything else you enter. The same model serves both programs, so there is nothing to describe twice — just add "Permeability" to each region.

```
spec = SeepageSpecFromModel[model, {14., 10.}]; (* upstream, downstream *)
sol  = SeepageUnconfined[spec];

sol["Converged"]          (* True *)
sol["WaterTableDaylights"] (* did it emerge on the face? *)
SeepagePlot[sol, "PorePressure"]
FlowNet[sol]

mat2  = Append[mat, "PorePressure" -> SeepagePoreSpec[sol]];
model2 = SlopeModel[geom, mat2, "StrengthFactor" -> 1.];
```

Rebuild the model. One already built does not know about the field.

On the two-layer slope of `Examples/02-Seepage.wl`:

no water	1.6742
computed seepage	1.3400
hand-drawn flat table at 13 m	1.1897

Water alone costs 20%. The hand-drawn table is 11% low — and which way it errs depends entirely on where you draw it, which is the argument for computing the field.

Two notes

`SeepageUnconfined` may print several `Part::partd` messages mentioning `NDSolve`FEM`MeshQualityEstimatorsDump`. They come from inside Wolfram's own mesh quality estimator, not from this package, and do not affect the solution. Quiet them if you like.

`TerraPercolatio` and `Limitas` both export a function called `PorePressureField`, with different arguments, so one shadows the other. Use **`SeepagePoreField`** and **`SeepagePoreSpec`** and the ambiguity never arises.

6 When a circle is the wrong shape

A thin, deep, weak stratum is exploited by a mechanism that runs *along* it — steep active wedge, long sliding base, passive wedge at the toe. No circle is that shape, and `NonCircularCritical` cannot travel from one to the other because every intermediate shape scores worse than both ends.

```
blk  = WeakLayerSurfaces[model];      (* systematic two-wedge blocks, seconds *)
resCK = CuckooCritical[model];        (* global search, stochastic, ~40 s *)

SurfaceLayerUsage[model, resJ]        (* did the circle enter the band? *)
resCK["LayerUsage"]                   (* did the cuckoo? *)
```

The cross-check, which is not optional

Both searches minimise Janbu, and Janbu satisfies force equilibrium only. A search free to choose any shape can find a surface that suits *Janbu's error* rather than the slope, and will report a satisfyingly low number when it does. "LayerUsage" does not catch this: it says whether the surface entered the band, not whether entering it helped.

```
MorgensternPriceFoS[model, resCK["Surface"]]["FoS"]
SpencerFoS[model, resCK["Surface"]]["FoS"]
SarmaFoS[model, resCK["Surface"], "Psi" -> "Sarma"] ["FoS"]
```

Compare against the same three on the critical circle. **If the surface is genuinely more critical, every method says so. If only Janbu says so, Janbu is what was optimised.** On the 500 mm band of `Examples/03-WeakLayer.wl` all four agree and the cuckoo surface is 13.5 % below the circle on Morgenstern–Price — a real mechanism, and one the circular search missed entirely, having put nothing at all in the band while converging cleanly and warning about nothing.

The rule. Search with Janbu because it is cheap. Re-evaluate the winner with Morgenstern–Price, Spencer and Sarma. Believe the improvement only if they all agree it is one. Report the rigorous factor on the surface you accept — never the Janbu factor the search happened to minimise.

The same map, for block mechanisms

`WeakLayerSurfaces` hands back an already-evaluated population, so it can be mapped with no sweep of its own:

```
blk = WeakLayerSurfaces[model];          (* 152 two-wedge blocks, ~1.3 s *)
sub = SubCriticalFromSurfaces[blk, 1.9];
CriticalityMapPlot[model, CriticalityField[model, sub]]
```

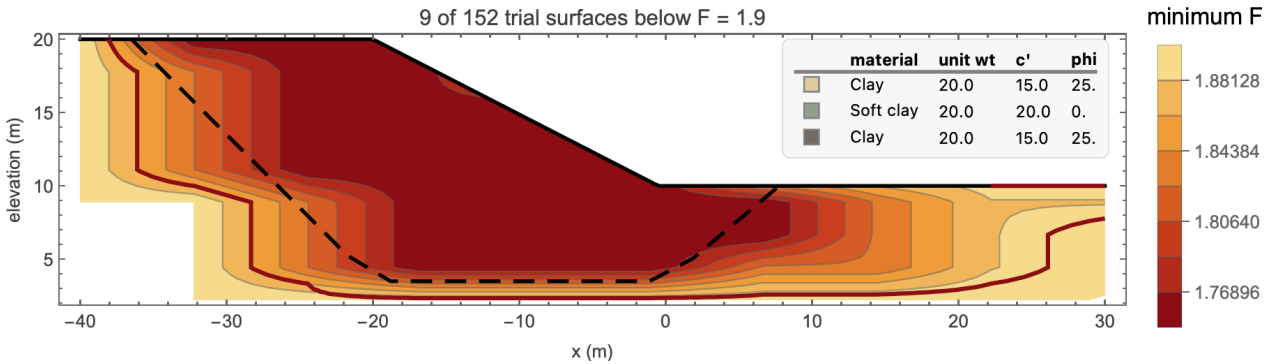


Figure 4: The 500 mm band of `Examples/03-WeakLayer.wl`, threshold 1.9. Nine of 152 block mechanisms qualify, the best at $F = 1.7502$. Dashed, that mechanism: active wedge down, base along the band, passive wedge up to the toe. "DeepestArc" returns 3.5 m — the map bottoms out on the band, which is the whole point of the example.

Do not map the cuckoo population. CuckooCritical returns one winner and a best-so-far trace, not a population; and even if every nest were kept, a stochastic search is built to *concentrate* its samples where it is converging. Mapped, that shows where the optimiser spent its time. Ground left blank would mean “never visited”, which reads exactly like “safe” and is not the same statement at all. `WeakLayerSurfaces` is a systematic grid over start points, end points and wedge angles, which is what makes the picture mean what it looks like it means. Search with the cuckoo; map with a systematic population.

7 Traps

1. **BishopCriticalCircle and JanbuCritical return "Circle", not "Surface".** Asking a circular result for "Surface" gives `Missing[]` and the call it is passed to returns *unevaluated* — which reads like a hang, not an error. Pass the whole result for a circle; pass `res["Surface"]` for a non-circular one.
2. **A trailing comma before `|>` makes an invalid Association and silently loses every key in it.**
3. **A missing comma between keys absorbs everything after it into the previous value.**
4. **Angles.** 30 is 30 radians.
5. **Stale models.** Rebuild after any `Get`, and after adding a pore pressure field.
6. **A water table polyline that stops short of the domain** is clamped to its end value, not extrapolated.
7. **A critical surface transcribed by hand is not the same surface.** Rounding node elevations to two decimals has moved F by more than 50% on a thin band. Pass the "Surface" value itself, never a printed copy.
8. **A search that converged is not a search that looked.**
9. **Two different quantities are both called "Lambda".** In Morgenstern–Price and Spencer it is the dimensionless ratio in $X = \lambda f(x)E$, about 0.42 and 0.34 on the QuickStart circle. In Sarma it scales the shear itself, $X_i = \lambda q_i$, so it is a *force* — 66.3 kN/m on that same circle. They are not interchangeable, and any code that reads `res["Lambda"]` without first asking which method produced it is wrong by a factor of two hundred.
10. **Sarma ignores horizontal surface loads.** Not just the crack thrust — *any* horizontal external force. The vertical component of a load reaches it through W ; the horizontal component never enters its equations. A purely horizontal 30 kPa traction on the QuickStart crest moves

Bishop $1.40646 \rightarrow 1.30876$, Janbu $1.41850 \rightarrow 1.27238$ and Morgenstern–Price $1.40415 \rightarrow 1.30649$, and leaves Sarma at 1.40406 , *unchanged to every digit*. **Do not use Sarma on a model with a horizontal surface load, a crack water thrust, or a seismic force applied as a traction** until this is fixed — which is unfortunate, because seismic work is exactly what Sarma is for. Its own K_c is unaffected; the fault is in external tractions, not in the acceleration it solves for.

11. **IntersliceForces on a *Sarma* result returns a zero-shear reference, not Sarma’s stress state.** Recovering Sarma’s own interslice forces needs the q parameterisation, which IntersliceForces does not implement, so it returns $X = 0$ deliberately. The tell is the force residual: 0.206 , the same as Bishop’s. Read it as a comparison baseline or not at all.

8 What is in the folder

Limitas.wl	the loader. This is the only file you Get
BishopSimplified.wl	the model, the slice builder, the circle grid
JanbuSimplified.wl	Janbu and the local non-circular optimiser
MorgensternPrice.wl, Spencer.wl, Sarma.wl	the rigorous methods
CuckooSearch.wl	the global non-circular search
TerraPercolatio.wl	the seepage solver
LimitasSeepage.wl	the bridge between the two
Criticality.wl	add-on: the sub-threshold map of sections 3.5 and 6
ThrustLine.wl	add-on: the line of thrust, section 4. Both add-ons are read by Limitas.wl; keep them in this folder
Examples/	three runnable files, in order

Limitations

1. The reported factor is an upper bound on the true minimum, typically 1–3% high at "Grid" $\rightarrow 12$.
2. Neither non-circular search offers a lower bound, and both minimise Janbu rather than the method you will quote.
3. Sarma is the 1973 vertical-slice formulation. Genuine wedge mechanisms (Sarma 1979) are not implemented.
4. No reinforcement. A tension crack is available (section 2.4), but only Bishop and Janbu apply its water thrust.
5. Finite element strength reduction and runout are Part 2, and are not in this folder.