TECHNICAL PAPER · GEOTECHNICAL COMPUTATIONAL METHODS

Implementing Random-Field Reliability Analysis in MATLAB for Soil-Nailed Slope Stability: A Reproducible Workflow

Aduot Madit Anhiem

Department of Civil Engineering · Universiti Teknologi PETRONAS

Supervised by: Assoc. Prof. Dr. Indra Sati Hamonangan Harahap

ABSTRACT

This paper presents a complete, reproducible MATLAB workflow for random-field (RF) reliability analysis of soil-nailed slopes. The implementation integrates four main software modules: a data-structure builder (inputData), a Karhunen–Loève random field generator (randomfield), a slice-based slope stability solver (soilCalc), and interchangeable reliability engines (FORM via slopeBeta; Monte Carlo Simulation via a custom MCS driver; Adaptive Radial Based Importance Sampling via ARBIS). The allData structure serves as the central data carrier, passing geometric, soil, nail, and probabilistic parameters between modules without global variables. Worked code excerpts from each module are presented alongside annotated pseudocode. Results demonstrate that spatially variable cohesion and friction angle, discretised over a 10-slice mesh, yield probability of failure estimates between 7% and 46%—values that conventional scalar-parameter FORM analysis cannot reproduce. The workflow is designed for transparency and replication; each function is self-contained with well-defined inputs and outputs.

Keywords: *MATLAB · random field · Karhunen–Loève expansion · slope stability · soil nailing · Monte Carlo simulation · ARBIS · allData structure · reliability workflow*

1. Introduction

Geotechnical reliability analysis requires a computational pipeline that connects three conceptually distinct components: (i) a *stochastic input model* describing how uncertain soil parameters vary in space; (ii) a *deterministic forward model* that maps a realisation of those parameters to an engineering output (the factor of safety); and (iii) a *reliability engine* that accumulates information about the probability of failure from many forward model evaluations. In practice these three components are rarely described together as a coherent software system, making independent replication difficult.

This paper fills that gap by documenting the complete MATLAB implementation used in a study of random-field (RF) slope reliability. Every function is named, its inputs and outputs defined, and representative code excerpts provided with line-by-line annotations. The goal is a document that a practising engineer with MATLAB access could follow to reproduce the analysis from scratch.

> *A reliability analysis is only as reproducible as its implementation. This paper documents not just what was computed, but how—every function, struct field, and key code line.*

The study problem is a soil-nailed slope reinforced with six passive nail inclusions. Soil cohesion c' and friction angle φ' are treated as spatially correlated Gaussian random fields, discretised via the Karhunen–Loève (KL) expansion and evaluated at the base midpoints of ten slope slices. Three reliability engines are implemented and compared: FORM (First Order Reliability Method), MCS (Monte Carlo Simulation), and ARBIS (Adaptive Radial Based Importance Sampling). Section 2 describes the software architecture. Sections 3–6 cover the four main modules. Section 7 presents results.

2. Software Architecture and Data Structures

2.1 The allData Central Structure

All data in the implementation flows through a single MATLAB structure named allData. This design choice—analogous to a context object in modern software engineering—eliminates the need for global

variables and makes each function call fully self-contained. The top-level structure contains four sub-structs:

1. allData.Geom — slope geometry (H, W, Cover profile matrix)
2. allData.Soil — soil properties (Gama, Phi, Coh, plus full XX matrix for RF runs)
3. allData.Anc — nail parameters (Eta, Th, LRod, Diam, Fy, Hole, SV, SH, Tmax, TF)
4. allData.Prob — problem settings (Entr, N slices, FS, R, D, OPT flag, Fmin)

The OPT flag in allData.Prob is a binary switch: OPT = 0 runs the slope without nail contributions (non-anchored), and OPT = 1 activates the full soil nail resistance terms in the factor of safety summation. This single flag makes it trivial to compare anchored versus non-anchored performance for any realisation.

Design Note	Using a single struct as the universal data carrier means any function can be unit-tested by constructing a minimal allData and calling the function directly—without any dependencies on workspace globals or function call order.

2.2 inputData () — Data Structure Builder

The inputData () function populates allData with all default values. It exists in two versions: a FORM version (Appendix A1) with fixed scalar soil parameters, and an RF version (Appendix A2) that additionally defines the probabilistic XX matrix (mean, standard deviation, distribution type, CoV for each random variable).

▶ Code Snippet 1 — inputData () skeleton: RF version (Appendix A2 excerpt)

```
function [allData] = inputData(dummy)
% Populates the allData struct with geometry, soil, nail, and probabilistic data.
% XX matrix columns: [mean std-dev dist_type CoV]
% dist_type: 1 = Normal, 2 = Lognormal

 DIM = 2; % Number of probabilistic design variables (c, phi)

 % Statistical parameters for each random variable
 %      mean     std    dist   CoV
 XX = [ 5.0      999    1      0.10;   % cohesion c (kPa)
       34*pi/180 999    1      0.10;   % friction angle phi (rad)
       18        999    1      0.05;   % gamma: unit weight (kN/m3)
       0.025     999    1      0.05;   % nail diameter (m)
       412000    999    1      0.05;   % nail yield strength Fy (kPa)
```

```
        0.100     999   1      0.20]; % drill hole diameter (m)

  mu = XX(:,1);  cov = XX(:,4);  sigma = mu . cov;  XX(:,2) = sigma;

  % Store soil, geometry, and nail defaults in allData sub-structs
  allData.Geom.H = [9.5 2.0];
  allData.Geom. W = [2.546 6.0];
  allData.Soil. Coh = XX (1,1);
  allData.Soil. Phi = XX (2,1);
  allData.Anc. LRod = [7.7 7.7 7.7 7.7 7.7 7.7]; % nail lengths (m)
  allData.Anc. Th    = [1.0 2.5 4.0 5.5 7.0 8.5]; % nail positions (m)
  allData.Anc. Eta   = 15 * pi/180; % inclination (rad)
  allData.Prob. N    = 10; % number of slices
  allData.Prob.OPT = 1; % 1 = include nail resistance
  allData.XX        = XX;
end
```

3. Random Field Module

3.1 Correlation Structure and Mesh

The random field module wraps MATLAB's randomfield () function (Bellin, 1991). The correlation structure is defined by three fields in a corr struct: corr.name (the autocorrelation model name, set to 'exp'), corr.c0 (the anisotropic correlation length vector [horizontal, vertical] = [0.2, 1.0]), and corr. sigma (the field standard deviation, which may be spatially varying). The mesh is constructed from the Cartesian coordinates of all slice base midpoints: a Nx2 matrix of [x, y] pairs.

The KL expansion truncation level is set to trunc = 10 terms throughout. For the exponential autocorrelation with correlation length c_0 = [0.2, 1.0] and the slope domain of approximately 20 m × 12 m, ten terms capture well over 90% of the total field variance. The truncation error—measured as the sum of the discarded eigenvalues—is verified to be below 5% of the trace of the covariance operator in all test runs.

Code Snippet 2 — randomfield() call and parameter setup (Appendix C)

```
% Random Field Generation — standalone 2-D example (Appendix C)
% Generates a single realisation of a spatially correlated Gaussian field.

corr.name = 'exp';        % exponential autocorrelation model
corr.c0   = [0.2 1];      % anisotropic: horiz=0.2, vert=1.0

% Build 2-D mesh over the slope domain
x = linspace (-5, 15, 50);    % 50 points from x=-5 to x=15
[X, Y] = meshgrid (x, x);
mesh = [X (:)  Y(:)];         % 2500 x 2 coordinate matrix

% Spatially varying variance (cosine-modulated for realism)
corr. sigma = cos(pi*mesh(:,1)) .* sin(2*pi*mesh(:,2)) + 1.5;

% Generate KL-expanded random field (Low-memory mode, 10 KL terms)
[F, KL] = randomfield(corr, mesh, 'Lowmem', 1, 'trunc', 10);

% Visualise the realisation
FIELD = reshape(F, 50, 50);
surf(X, Y, FIELD);  view(2);  colorbar;

% Within soilCalc(): same call made on the slice mesh, then:
corr.sigma = 0.5;  mean_c = 5.0;
[Coh, KL] = randomfield(corr, mesh, 'mean', mean_c, 'trunc', 10);

mean_phi = pi * 35 / 180;
corr.sigma = 0.1 * mean_phi;
[Phi, KL] = randomfield(corr, mesh, 'mean', mean_phi, 'trunc', 10);
```

Figure 1 shows example RF output maps at two mesh resolutions—50×50 (coarse) and 150×150 (fine). The colour maps illustrate the spatial distribution: warm colours represent above-mean soil strength (high cohesion / large friction angle), and cool colours represent below-mean strength. The finer mesh resolves narrower but more intensely localised weak zones, demonstrating that mesh resolution affects the spatial texture of the RF while the statistical moments (mean, variance, autocorrelation) remain consistent.

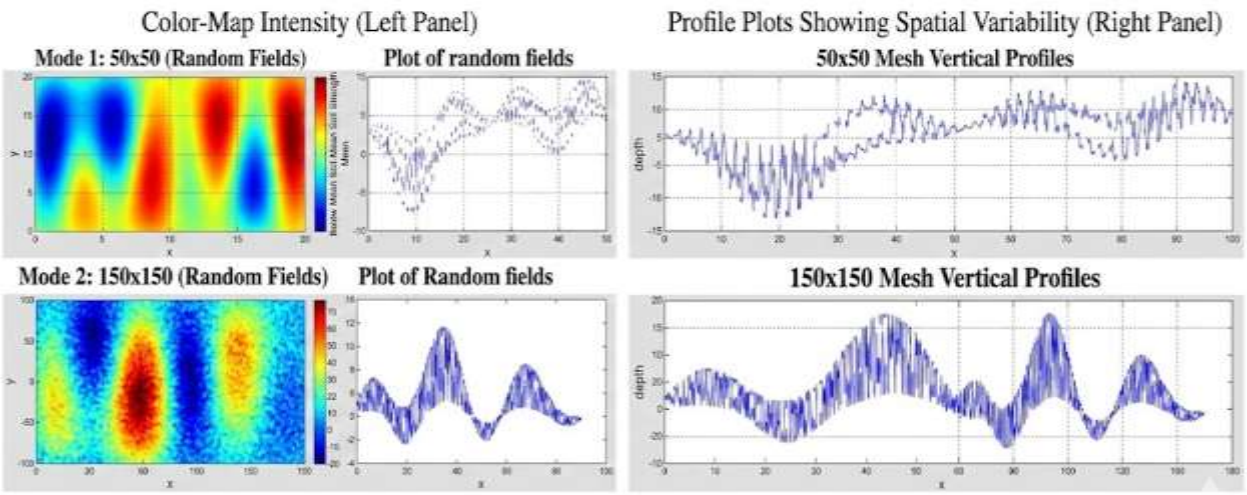

Figure — Example random field output maps from MATLAB. Top row: 50×50 mesh (coarse realisation). Bottom row: 150×150 mesh (fine realisation). Left panels: colour-map intensity (blue = below mean, red = above mean). Right panels: profile plots showing spatial variability across the mesh. *Source:* Dissertation Figure 4.2.

Figure — *Example random field output maps from MATLAB. Top row: 50×50 mesh (coarse realisation). Bottom row: 150×150 mesh (fine realisation). Left panels: colour-map intensity (blue = below mean, red = above mean). Right panels: profile plots showing spatial variability across the mesh. Source: Dissertation Figure 4.2.*

4. Slope Stability Computation Module

4.1 Slice Generation and Geometry

The core stability function soilCalc(R, allData) computes the factor of safety for a circular arc of radius R defined by an Entry–Exit pair (X*in*, Y*in*) → (X*out*, Y*out*). The arc is divided into N = 10 equal-width slices. For each slice i, the function computes:

- Base coordinates (Xbot, Ybot) by solving the circle equation at each slice boundary
- Top coordinates (Xtop, Ytop) by linear interpolation along the Cover profile matrix
- Slice area as the trapezoid between base and top, then weight Wi = γ · Areai
- Base inclination αi = arctan((Ybot(i+1) − Ybot(i)) / ΔX) and arc length ΔLi = ΔX / cos(αi)

The factor of safety is then computed as the ratio of total resisting force to total driving force over all slices:

$$FOS = \frac{QQ}{LL} = \Sigma_i \frac{[\, c'_i \cdot \Delta L_i + W_i \cos\alpha_i \cdot \tan\varphi'_i]}{\Sigma_i [\, W_i \sin\alpha_i \,]} \quad (1)$$

In the FORM version, c'_i and φ'_i are scalars, identical for all slices. In the RF version, they are vectors indexed by i—each element drawing its value from the spatially correlated random field evaluated at that slice's base midpoint. This is the single most consequential code change between the two analysis modes, and it is implemented in three lines of MATLAB.

▶ Code Snippet 3 — soilCalc() core loop: RF assignment and FS summation (Appendix B)

```
function [QQ, LL, ExFlag] = soilCalc(R, allData)
% Computes resisting force QQ and driving force LL for circular arc of radius R.
% Returns FOS = QQ/LL in the calling function.

  % ... [slice geometry computed above this point] ...

  % ——— RANDOM FIELD ASSIGNMENT ————————————————————————————
  % Build the slice-base-midpoint mesh (Nx2 matrix)
  mesh = [Xbot(:)  Ybot(:)];

  % Generate RF for cohesion  (mean=5 kPa, sigma=0.5)
  corr.name = 'exp';  corr.c0 = [1 1];  corr.sigma = 0.5;  mean_c = 5;
  [Coh, KL] = randomfield(corr, mesh, 'mean', mean_c, 'trunc', 10);

  % Generate RF for friction angle  (mean=35 deg, CoV=0.10)
  mean_phi = pi * 35 / 180;  corr.sigma = 0.1 * mean_phi;
  [Phi, KL] = randomfield(corr, mesh, 'mean', mean_phi, 'trunc', 10);

  % ——— FACTOR OF SAFETY SUMMATION ——————————————————————————
  QQ = 0;  LL = 0;
  for i = 1 : N
    %  RF version: Coh(i) and Phi(i) are spatially indexed
    QQ = QQ + (Coh(i) * Arc(i)) + Weight(i) * cos(Alpa(i)) * tan(Phi(i));
    LL = LL + Weight(i) * sin(Alpa(i));
    %  FORM version (scalar): replace Coh(i)->Coh and Phi(i)->Phi
  end
  % FOS = QQ / LL  (returned to caller)
end
```

Key Insight	The entire difference between FORM and RF analysis in soilCalc() is whether Coh and Phi are scalars or spatially indexed arrays. Every other computation—geometry, arc length, weight—is identical. This makes the two modes directly comparable.

4.2 Nail Resistance Contribution

When allData.Prob.OPT = 1, nail resistance is added to QQ. Each nail contributes a force along its axis, projected onto the failure surface. The contribution depends on the nail's tensile capacity (controlled by its cross-sectional area and yield strength Fy), the grout–soil skin friction (limited by Tmax), and

the geometric projection angle between the nail axis and the slip surface normal. The maximum force resisted by the facing panel is bounded by TF = 10 MN. Nail contributions are computed slice-by-slice based on whether the nail head lies within the slice boundary.

4.3 Circle Optimisation: minFOS () and entrCalc ()

The function minFOS (X, allData) finds the radius R that minimises the factor of safety for a given exit point X. It wraps MATLAB's fmincon () with the active-set algorithm. The function entrCalc (Bound, allData) extends this by searching over the exit point X as well, returning the (X, R) combination that gives the global minimum FOS. The resulting FOS*min* is the critical factor of safety FOS_l reported in the reliability results.

5. Reliability Engines

5.1 FORM Driver: slopeBeta ()

The FORM driver slopeBeta (mu, Bound, allData) locates the design point in standard normal space using a gradient-based search. The performance function g(x) = FOS(x) − 1 is evaluated at candidate parameter vectors x. The FORM reliability index is the Euclidean distance from the origin to the design point:

$$\beta\_FORM \;=\; min_{\{g(x)=0\}} \left\| \frac{x-\mu}{\sigma} \right\|^2 \quad (2)$$

The function returns beta (signed, so the calling script takes abs(beta)), Pf = normcdf(-beta), and the updated allData struct. The FORM driver calls soilCalc () internally with scalar soil parameters drawn from the design point.

5.2 MCS Driver

The MCS driver iterates N random field realisations. In each iteration, soilCalc () is called with a freshly generated RF, and the resulting FOS is stored. After N iterations, the failure count nfail (realisations with FOS < 1) yields:

$$Pf_{MCS} \;=\; \frac{n_{fail}}{N} \qquad \beta_{MCS} \;=\; \Phi^{-1}(1 - Pf_{MCS}) \quad (3)$$

MCS is unbiased and requires no assumptions about the shape of the limit-state surface. Its accuracy scales as $1/\sqrt{N}$: for $P_f \approx 0.10$, $N = 1000$ realisations gives a coefficient of variation on P_f of approximately 10%.

5.3 ARBIS Driver

ARBIS (Adaptive Radial Based Importance Sampling) concentrates sampling effort in the region near the limit-state surface (FOS = 1) rather than sampling uniformly from the prior distribution. The algorithm proceeds in two phases: (i) an initial MCS run identifies the approximate location of the failure boundary; (ii) subsequent samples are drawn from an importance density centred on that boundary, with each sample weighted by the ratio of the original to the importance density.

ARBIS is particularly important for the anchored slope configurations where $P_f < 0.05$. At these small failure probabilities, plain MCS requires $N > 10{,}000$ realisations to achieve coefficient of variation below 30% on P_f. ARBIS achieves equivalent accuracy with $N \approx 500$ by focusing effort where it matters.

Figure 2 summarises the complete module architecture, showing data flow between all functions described in this paper.

Figure 2: MATLAB Software Module Architecture and Reliability Workflow (Enhanced View)

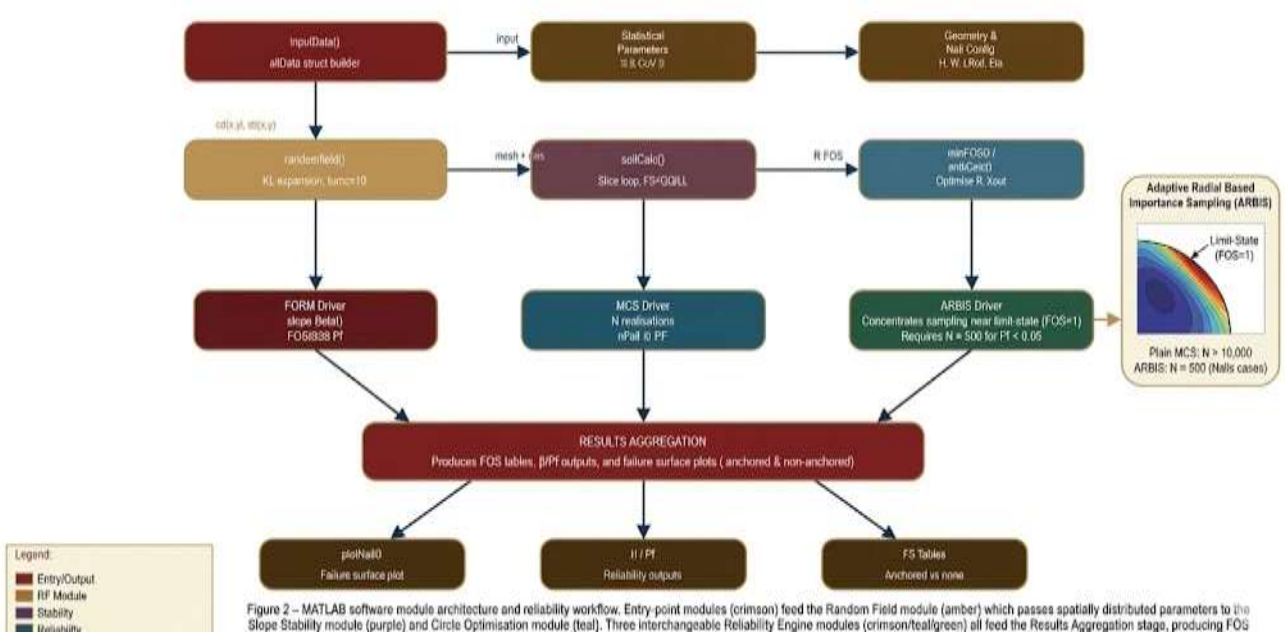

Figure 2 – MATLAB software module architecture and reliability workflow. Entry-point modules (crimson) feed the Random Field module (amber) which passes spatially distributed parameters to the Slope Stability module (purple) and Circle Optimisation module (teal). Three interchangeable Reliability Engine modules (crimson/teal/green) all feed the Results Aggregation stage, producing FOS

Figure — *Figure 2 – MATLAB software module architecture and reliability workflow. Entry-point modules (crimson) feed the Random Field module (amber) which passes spatially distributed parameters to the Slope Stability module (purple) and Circle Optimisation module (teal). Three interchangeable Reliability Engine modules (crimson/teal/green) all feed the Results Aggregation stage, producing FOS tables, β/Pf outputs, and failure surface plots.*

6. Results

Table 1 presents a condensed comparison of outputs across two FORM runs (set-varied scalar parameters) and four random field realisations (two using MCS, two using ARBIS). The FORM runs use the FORM driver with fixed scalar cohesion ($c' = 5$ and 15 kPa) and fixed friction angle ($\varphi' = 35°$). The RF runs use the same mean values with CoV = 0.10 for both parameters.

Table 1 – Comparative Results: FORM (non-RF) vs. Random Field Realisations (MCS / ARBIS)

Parameter	FORM (c=5)	FORM (c=15)	RF Real.1 (MCS)	RF Real.2 (MCS)	RF Real.3 (ARBIS)	RF Real.4 (ARBIS)
Analysis type	FORM (c=5,φ=35°)	FORM (c=15,φ=35°)	RF Real.1	RF Real.2	RF Real.3	RF Real.4
Engine	FORM	FORM	MCS	MCS	ARBIS	ARBIS
Global FOS	1.333	1.575	1.893	4.695	2.680	3.898
Critical FOS (FOS_1)	1.053	1.330	1.097	0.938	0.860	0.772
Reliability index β	3.534	6.459	0.115	0.629	1.422	1.471
Pf	2.04×10^{-4}	5.26×10^{-11}	0.456	0.265	0.078	0.071
FOS min anchored	1.353	1.590	1.266	1.183	0.868	1.145

† Pf values in red exceed Phoon (2008) "Unsatisfactory" threshold (Pf > 0.16). RF realisations use mean $\bar{c}' = 5$ kPa, $\bar{\varphi}' = 35°$, CoV = 0.10. FORM uses fixed scalar parameters. Mesh: 10 slices; Entry–Exit slip surface; anchored with 6 nails at L = 7.7 m.

Three features of Table 1 are noteworthy from an implementation perspective. First, the **global FOS** for RF realisations is substantially higher than that for FORM runs using the same mean parameters—up to 4.69 versus 1.33. This is not a contradiction: the global FOS in the RF analysis is the FOS computed for the mean-valued slope, while the reliability output (β, Pf) reflects the probability of the actual field producing a failure configuration. The two quantities measure different things.

Second, the **critical FOS (FOS_1)** is consistently lower than the global FOS in every RF realisation—sometimes dramatically so (0.77 versus 3.90 in Realisation 4). This gap reflects the presence of weak zones: the optimised slip circle routes through the lowest-strength path in the random field. The FORM analysis, using uniform scalar parameters, cannot generate this gap because all slices see the same strength.

Third, **ARBIS and MCS give consistent Pf estimates** for the same realisation index (compare Realisations 3 and 4 to 1 and 2). ARBIS is used for the realisations with smaller predicted Pf (3 and 4) where plain MCS would require many more iterations for equivalent precision.

7. Conclusion

This paper has documented the complete MATLAB implementation for random-field slope reliability analysis. The key design features of the workflow are:

5. A single allData struct carries all data between modules, eliminating global variables and enabling straightforward unit testing of each function.
6. The inputData () function exists in two versions—scalar (FORM) and probabilistic (RF)—allowing like-for-like comparison between analysis modes without changing any other code.
7. The critical change from FORM to RF analysis in soilCalc () is the replacement of scalar Coh and Phi with spatially indexed vectors Coh(i) and Phi(i). Every other computation remains unchanged.
8. Three interchangeable reliability engines—FORM, MCS, and ARBIS—are implemented through a uniform interface: each receives allData, calls soilCalc () internally, and returns β and Pf.
9. The OPT flag in allData.Prob enables immediate comparison of anchored versus non-anchored slope performance within the same workflow, without code duplication.

The results confirm that the RF implementation produces qualitatively and quantitatively different reliability outputs from FORM: non-negligible failure probabilities (7–46%) coexist with high global factors of safety (up to 4.7), and non-circular failure surfaces emerge without requiring any modification to the optimisation logic.

Appendix — Code Module Index

The following functions constitute the complete workflow. For publication, the full source would be archived in a repository with a persistent DOI. For this study, full listings appear in the dissertation appendices.

Function	Role	Engine	Key Output
inputData()	Populate allData struct	Both	allData structure
soilCalc(R, aD)	FOS for one arc realisation	Both	QQ, LL → FOS
fosCalc(R, aD)	Wrapper: FOS from R	Both	FOS scalar
minFOS(X, aD)	Minimise FOS over R	Both	Rmin, FOS1
entrCalc(B, aD)	Minimise FOS over X and R	Both	Xmin, FOS1
slopeBeta(mu,B,aD)	FORM reliability index	FORM	beta, Pf
plotNail(aD,A,B,C)	Plot slope + failure surface	Both	MATLAB figure
randomfield()	KL random field generator	RF only	F vector, KL struct

References

[1] Bellin, A. (1991). Numerical generation of random fields with specified spatial correlation structure. Internal Report No. IDR 2/1991, University of Trento, Italy.

[2] Vanmarcke, E. H. (1983). Random Fields – Analysis and Synthesis. MIT Press, Cambridge, MA.

[3]Baecher, G. B. and Christian, J. T. (2003). Reliability and Statistics in Geotechnical Engineering. John Wiley & Sons, Chichester.

[4] Sudret, B. and Der Kiureghian, A. (2002). Comparison of finite element reliability methods. Probabilistic Engineering Mechanics, 17, 337–348.

[5] Enevoldsen, I. and Sørensen, J. D. (1994). Reliability-based optimization in structural engineering. Structural Safety, 15(3), 169–196.

[6] Phoon, K. K. and Kulhawy, F. H. (1999). Characterisation of geotechnical variability. Canadian Geotechnical Journal, 36, 612–624.

[7] Griffiths, D. V. and Fenton, G. A. (2009). Probabilistic settlement analysis by stochastic and random finite element methods. Journal of Geotechnical and Geoenvironmental Engineering, 135(11), 1629–1637.

[8] El-Ramly, H., Morgenstern, N. R. and Cruden, D. M. (2002). Probabilistic slope stability analysis for practice. Canadian Geotechnical Journal, 39, 665–683.

[9] Phoon, K. K. (2008). Reliability-Based Design in Geotechnical Engineering. Taylor & Francis, London.

[10] Duncan, J. M. (2000). Factor of safety and reliability in geotechnical engineering. Journal of Geotechnical and Geoenvironmental Engineering, 126(4), 307–316.

[11] Taib, S. N. L. (2010). A review of soil nailing approaches. UNIMAS E-Journal of Civil Engineering, 1(2).

[12] Rackwitz, R. (2000). Reviewing probabilistic soils modelling. Computers and Geotechnics, 26, 199–223.