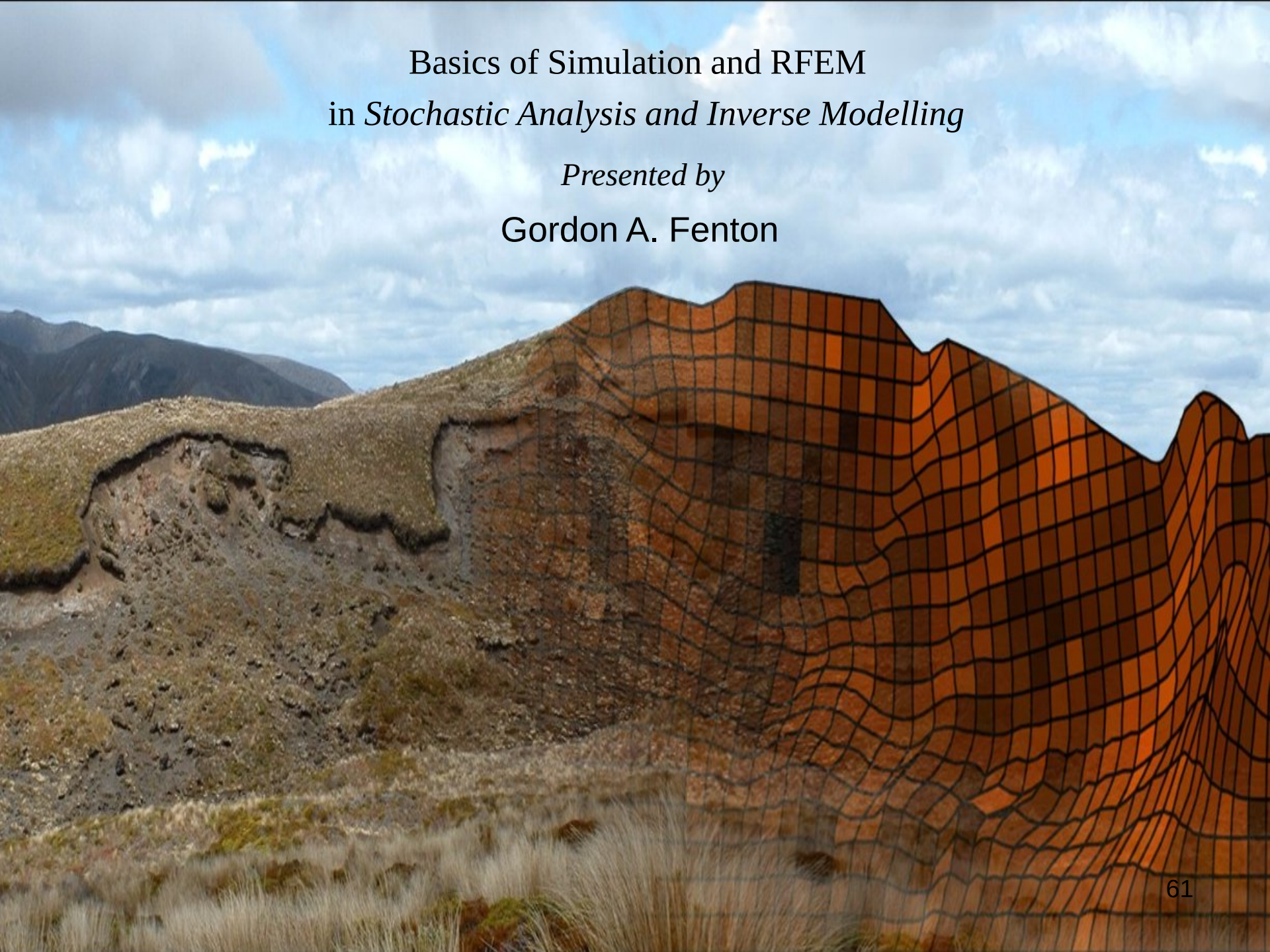
Basics of Simulation and RFEM

in *Stochastic Analysis and Inverse Modelling*

Presented by

Gordon A. Fenton



Introduction

- Many reliability questions are far too complex to determine analytically
- In this lecture we will investigate the technique of solving reliability assessment questions via Monte Carlo simulation
- We will start by looking at how to generate numbers which appear to be uniformly distributed between 0 and 1.
- We will then consider how to transform the U(0,1) variates into variates from arbitrary distributions (e.g. normal)
- Finally we will discuss how random fields are simulated.

Monte Carlo Simulation

Consider the problem of estimating P[$X < a$], where X is a complex function of other random variables. For example, suppose that

$$X = c\left(\frac{\tan^2\left(\frac{\pi}{4}+\frac{\phi}{2}\right)e^{\pi\tan\phi}-1}{\tan\phi}\right)$$

where c and ϕ are random. The analytical determination of the distribution of X is prohibitively difficult.

One simple solution is to simulate a series of possible realizations for both c and ϕ , compute the "response", X, for each and then assess the statistics of the resulting series of realizations of X;

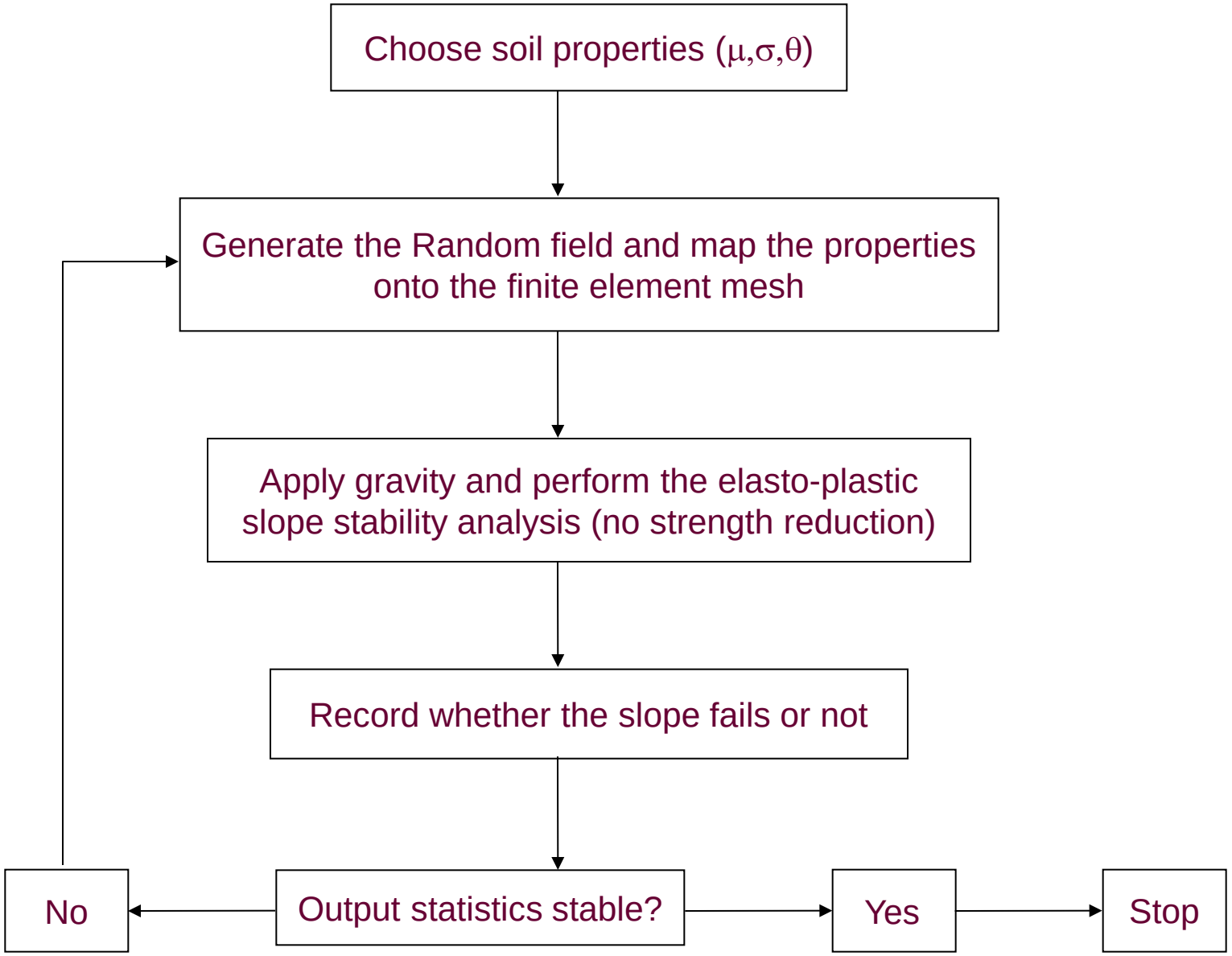
$$\mu_X \approx \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad\qquad \sigma_X^2 \approx \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

also $$\mathrm{P}[X > a] \approx \frac{\text{number of realizations where } X_i > a}{\text{total number of realizations}}$$

Monte Carlo Simulation

- Monte Carlo Simulation (MCS) involves
  - generating a realization of a random variable (or set of random variables) according to its prescribed distribution
  - computing the 'system' response
  - repeating the above two steps many times to assess probabilities/statistics (accuracy increases as the number of realizations increases)
- MCS is sometimes computationally intensive
- MCS has the advantage over first- and second-order methods of being able to estimate the entire distribution of the response.

Example: Monte-Carlo Simulation of Slope Stability



Accuracy of Monte Carlo Simulation Estimates

- each realization is an independent "observation" of a trial
- to estimate the probability of failure, we compute

$$\hat{p} = \frac{N_n}{n}$$

where N_n = number of trials which failed, and
n = total number of trials (realizations)

- since N_n follows a binomial distribution, we have

$$\mathrm{E}\left[\hat{p}\right] = \frac{\mathrm{E}\left[N_n\right]}{n} = \frac{np}{n} = p \qquad \text{(unbiased)}$$

$$\sigma_{\hat{p}} = \sqrt{\frac{\mathrm{Var}\left[N_n\right]}{n^2}} = \sqrt{\frac{npq}{n^2}} = \sqrt{\frac{pq}{n}} \simeq \sqrt{\frac{\hat{p}\hat{q}}{n}}$$

Accuracy of Monte Carlo Simulation Estimates

- If our maximum tolerable error on the estimated probability of failure, *p*, at 95 % confidence is *e*, then we must produce at least *n* realizations, where

$$n = \frac{z_\alpha^2 \hat{p}\hat{q}}{e^2} \simeq \frac{4\hat{p}\hat{q}}{e^2}, \qquad \text{where } \hat{q} = 1 - \hat{p}$$

- For example, if we expect the probability of failure to be around 0.001 and we want to estimate the probability of failure to an accuracy of better than 0.0001 (e.g. a relative error of 10%) with 95% confidence, then we require

$$n \simeq \frac{4(0.001)(0.999)}{(0.0001)^2} \simeq 400,000$$

realizations.

Accuracy of Monte Carlo Simulation Estimates

Since we are generally unable to achieve highly accurate probability estimates for low probability events, we often rely on

- distribution fitting to estimate probabilities in the tails – this assumes that there is some "persistence" in the relationship between the distributions of the input and response – since the response of geotechnical systems can often be approximated by a sum, or product, the central limit theorem suggests that fitting a distribution is appropriate.
- variance-reduction techniques, where the variance of the response is reduced through simulation "tricks", e.g. use *antithetic variates* (which are negatively correlated) to improve the accuracy of the estimate.

Pseudo-Random Number Generators

- The uniform distribution on the interval (0,1) is the simplest of distributions – it says that all numbers in the interval are equally likely to turn up.
- The most basic random number generator is one which "generates" random values that are uniformly distributed between 0 and 1.
- As we shall see, the numbers are not at all truly random, they just *appear* to be random – hence we call these *pseudo-random number generators*.
- The current standard in uniform random number generators are *arithmetic generators* in which each number is determined by one or more of its predecessors according to a fixed mathematical formula

Linear Congruential Generators

Linear Congruential Generators (LCGs) are currently the most popular type of arithmetic generators. In this method, a sequence of integers, Z_1, Z_2, … are defined by the recursive formula

$$Z_i = (aZ_{i-1} + c)(\mathrm{mod}\, m)$$

where m = modulus, a is a multiplier, c is an increment, and all three parameters are non-negative integers.

Example: what are the first three "random" numbers produced by the LCG

$$Z_i = (25Z_{i-1} + 55)(\mathrm{mod}\ 96)$$

using the starting *seed* $Z_0 = 21$?

Linear Congruential Generators

Solution: with $Z_0 = 21$, we get

$$Z_1 = (25(21) + 55)(\text{mod } 96) = 580(\text{mod } 96) = 4$$

$$Z_2 = (25(4) + 55)(\text{mod } 96) = 155(\text{mod } 96) = 59$$

$$Z_3 = (25(59) + 55)(\text{mod } 96) = 1530(\text{mod } 96) = 90$$

Comments:

- The modulus defines how many possible different random numbers can be produced by the LCG. For example, the above LCG can produce 96 different "random" numbers: 0, 1, …, 95.
- If we divide each Z_i by the modulus, m, the result lies in the interval [0,1).

Linear Congruential Generators

$$Z_i = \left(aZ_{i-1} + c\right)\left(\bmod m\right) \quad \rightarrow \quad U_i = \frac{Z_i}{m}$$

- Z_i will be a number from 0 to $m - 1$
- U_i will be a number from 0 to $(m - 1)/m$ on the interval [0,1)
- U_i divides the unit interval up into m possible values – in order for this to appear continuous, m should be large.
- a, c, and Z_0 should all be less than m.
- the sequence of Z_i are actually completely dependent, but they appear to be independent to most tests.
- the LCG can reproduce a stream of numbers simply by starting with the same seed, Z_0.

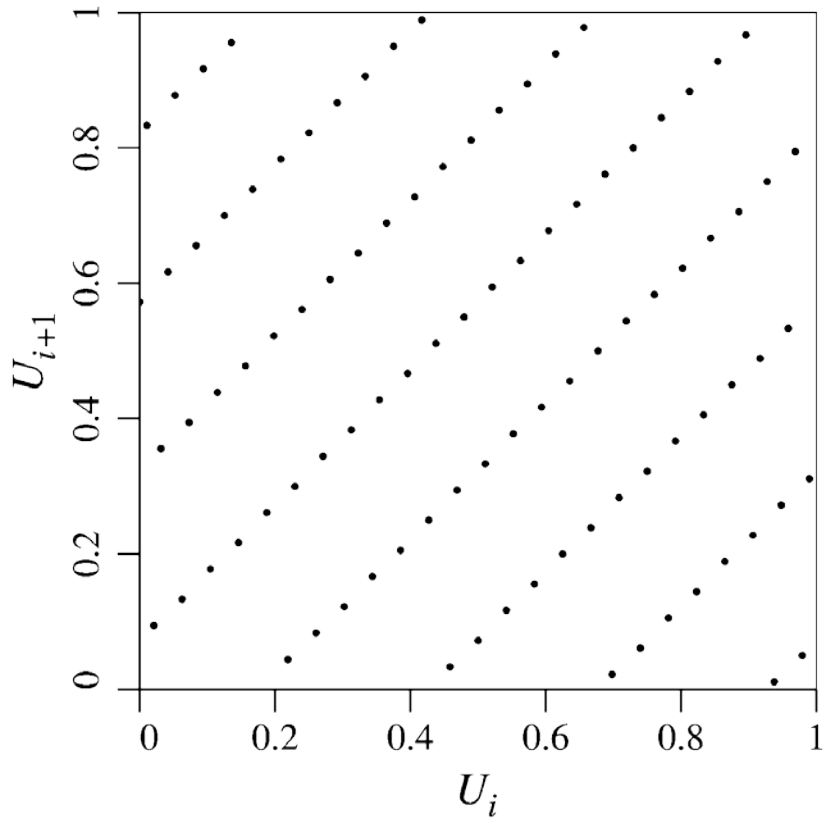
Linear Congruential Generators

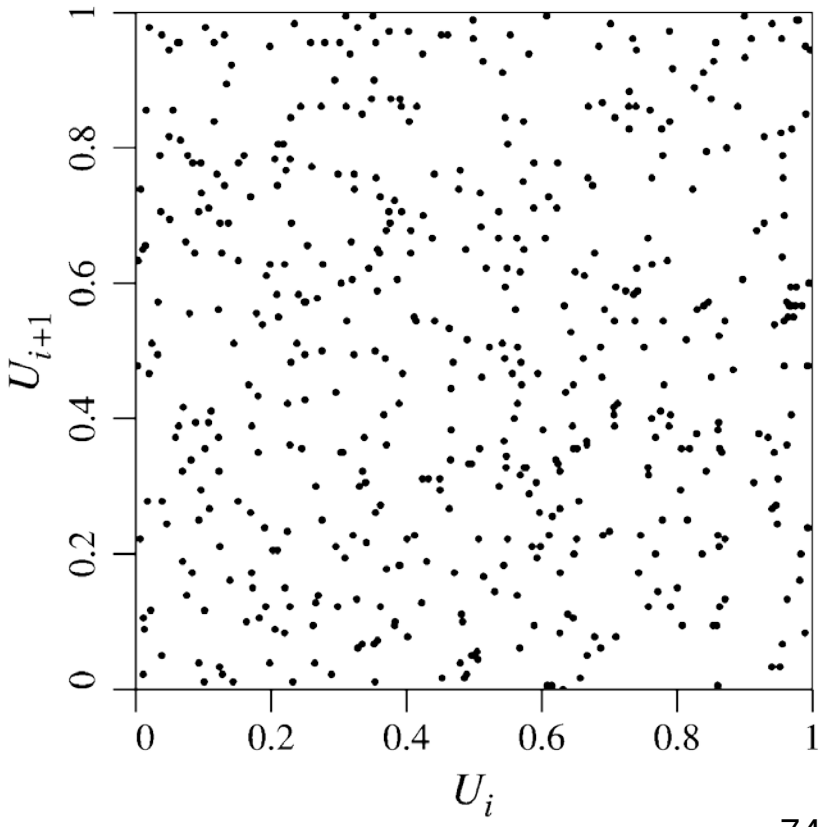
Periodicity:

- If $Z_0 = 3$ produces $Z_1 = 746$, then whenever $Z_{i-1} = 3$, the next number will be $Z_i = 746$.
- This property leads to an undesirable phenomenon called *periodicity* that quite a number of rather common generators suffer from (actually, all generators have maximum periodicity m, but some have periodicity much less than m).
- Suppose that you are unlucky enough to select the starting seed $Z_0 = 83$ that just happens to yield remainder 83 when $(83a + c)$ is divided by m. Then $Z_1 = 83$. In fact, your "random" sequence will be {83, 83, 83, …}. This particular stream of random numbers has periodicity equal to 1.

Testing Random Number Generators

Below we have plotted the pairs (U_i, U_{i+1}) for *Numerical Recipes* RAN2 generator (right) and the LCG $Z_i = (25Z_{i-1} + 55) (\text{mod } 96)$ (left)





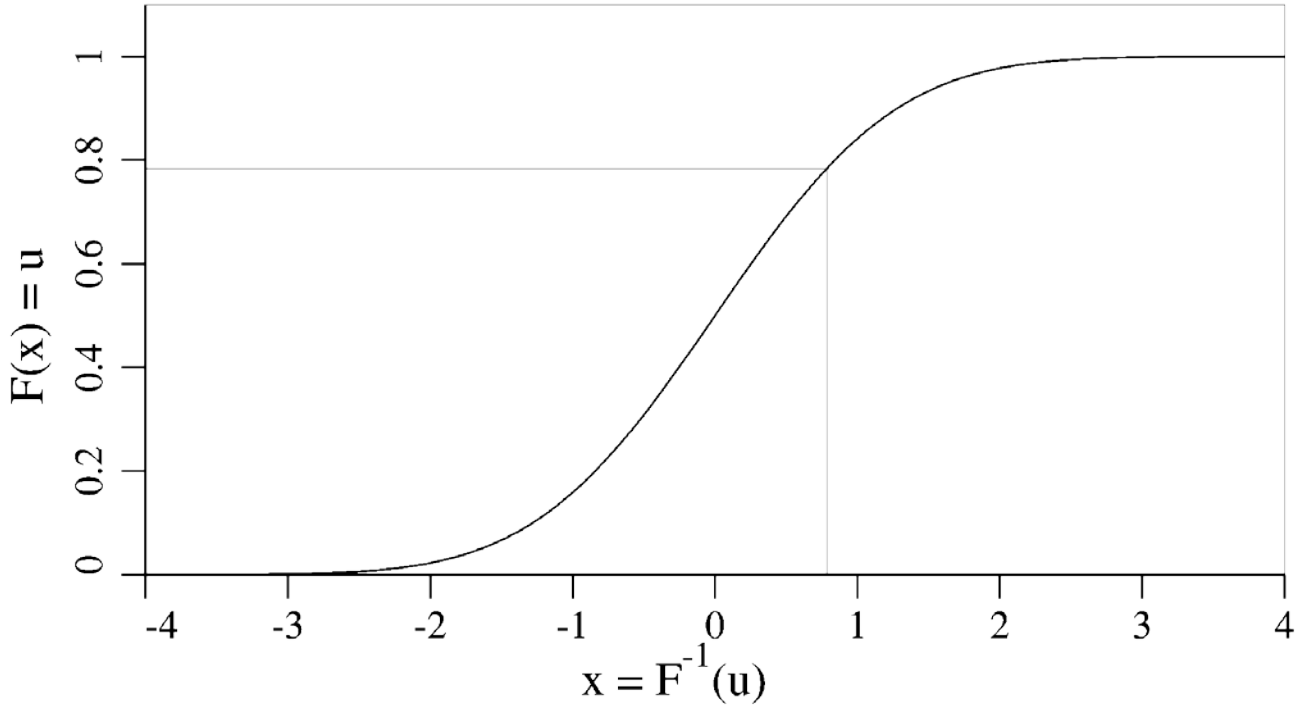
Generating Non-Uniform Random Variables

- The basic initial ingredient needed is a good U(0,1) generator.
- For most common distributions, efficient and exact generation algorithms exist which should be used.
- Three common methods of generating non-uniform random variables exist:
  - inverse transform
  - convolution
  - acceptance-rejection
- of these, the inverse transform and convolution methods are exact, while the acceptance-rejection method is only asymptotically exact (and computationally wasteful).

Inverse Transform Method

The basic idea of the inverse transform method is that since U_i is uniformly distributed between 0 and 1, it could represent a *probability*.

Setting $U_i = P[\ X < x_i\ ]$ and calculating the value of x_i which satisfies this equation gives a way of simulating realizations of X.



Inverse Transform Method: Example

Suppose that undersea slopes in the Baltic Sea fail at a mean rate of one every 400 years, and that the time between failures are independent and exponentially distributed. Generate two possible inter-failure times.

Solution: we start by generating two realizations of a uniformly distributed random variable on (0,1): say $u_1 = 0.27$ and $u_2 = 0.64$.

We know that for the exponential distribution

$$F(x) = 1 - e^{-\lambda x}$$

where $\lambda = 1/400$ in this case. Setting $u = F(x)$ and inverting gives

$$x = \frac{-\ln(1-u)}{\lambda}$$

Inverse Transform Method: Example

We note that $(1 - U)$ has exactly the same distribution as (U) so that realizations of X can be also computed from

$$x = \frac{-\ln(u)}{\lambda}$$

Using this form, we get the following two possible realizations of the inter-failure times

$$x_1 = -\frac{\ln(u_1)}{\lambda} = -400\ln(0.27) = 523 \text{ years}$$

$$x_2 = -\frac{\ln(u_2)}{\lambda} = -400\ln(0.64) = 179 \text{ years}$$

Inverse Transform Method

Advantages:

1. exact, efficient, and intuitive,
2. can be easily modified to generate from truncated distributions
3. can be modified to generate order statistics
4. facilitates variance-reduction techniques (where portions of the CDF are 'polled' more heavily than others, usually in the tails of the distribution, and then resulting statistics corrected to account for the biased 'polling'

Disadvantage:

1. Requires a formula for F^{-1}, which is not easily available for some distributions (e.g. normal, lognormal, beta, Gamma)

Special Case: Normal Distribution

The inverse CDF for the normal distribution involves solving the following for x,

$$u = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left\{ -\frac{1}{2}\left(\frac{s-\mu}{\sigma} \right)^2 \right\} ds$$

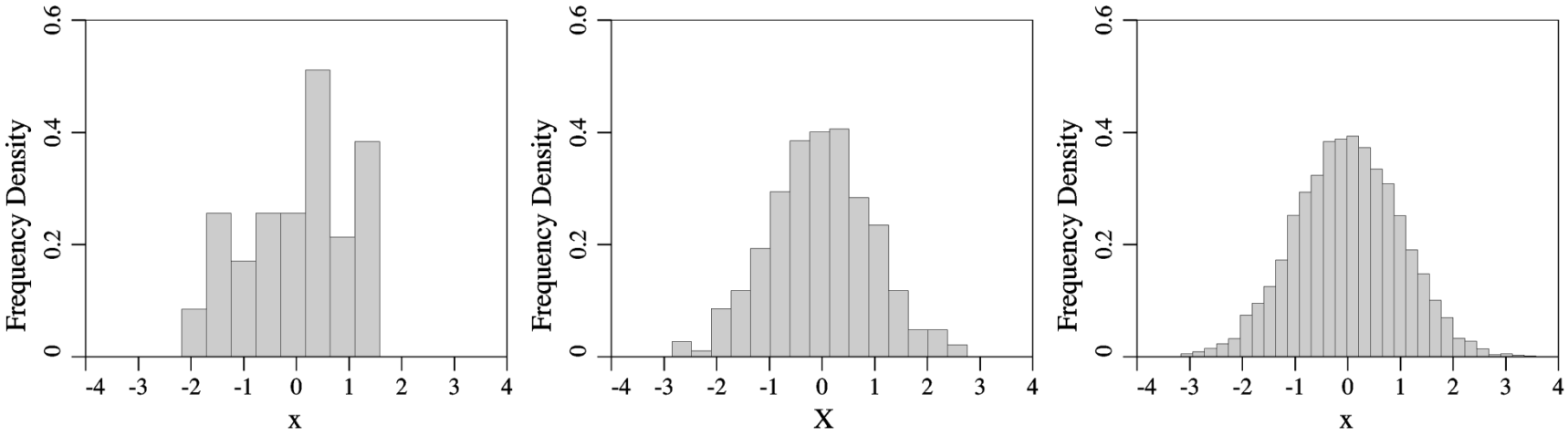
which must be done numerically. Box and Muller (1958) came up with an alternative way of generating standard normally distributed variates (zero mean, unit variance) using a radial transformation:

1. generate u_1 and u_2 from U(0,1)
2. form $x_1 = \sqrt{-2\ln u_1}\cos(2\pi u_2)$ and $x_2 = \sqrt{-2\ln u_1}\sin(2\pi u_2)$

This is exact, very simple, and efficient (for each pair of u's we get a pair of independent $N(0,1)$ random variates).

Example

The plots below show histograms of a standard normal *simulation* using n = 50, 500, and 5000.



n = 50 n = 500 n = 5000

Other Common Simulation Algorithms

Uniform on (a, b):

1. generate u from U(0,1)
2. return $x = a + (b - a)u$

Exponential:

1. generate u from U(0,1)
2. return $x = -\ln(u)/\lambda$

Weibull: $F(x) = 1 - e^{-(\lambda x)^{\beta}}$

1. generate u from U(0,1)
2. return $x = (-\ln u)^{1/\beta} / \lambda$

Lognormal:

1. generate normally distributed Y from N($\mu_{\ln X}$, $\sigma_{\ln X}$)
2. return $X = e^{Y}$

Simulation

Most spread-sheet programs include random number generators, often of questionable accuracy. Compilers include random number generators, also of questionable accuracy.

For serious simulation problems, get a good uniform random number generator (or at least ensure that your application makes use of one of these better generators). In particular, watch out for "periodicity" (where the random numbers start repeating after a bit – this is a Monte Carlo disaster!). See the 2nd Edition of Numerical Recipe's Ran2 program for a good generator (but avoid 1st Ed. version!).

Simulation of Random Fields

Several methods are available:

1) Covariance Matrix Decomposition:

Given a sequence of points in the random field, $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}^T$ then the values of **X** can be simulated using

$$\mathbf{X} = \underset{\sim}{\mu} + \mathbf{LG}$$

where $\underset{\sim}{\mu}$ is the mean at each point in the field,

G is a vector of *n* independent zero mean, unit variance, normally distributed random variables,

L is a lower triangular matrix satisfying $\mathbf{LL}^T = \mathbf{C}$

C is the matrix of covariances, Cov[X_i , X_j]

L is sometimes referred to as the square root of **C**. It is commonly computed using a Cholesky Decomposition.

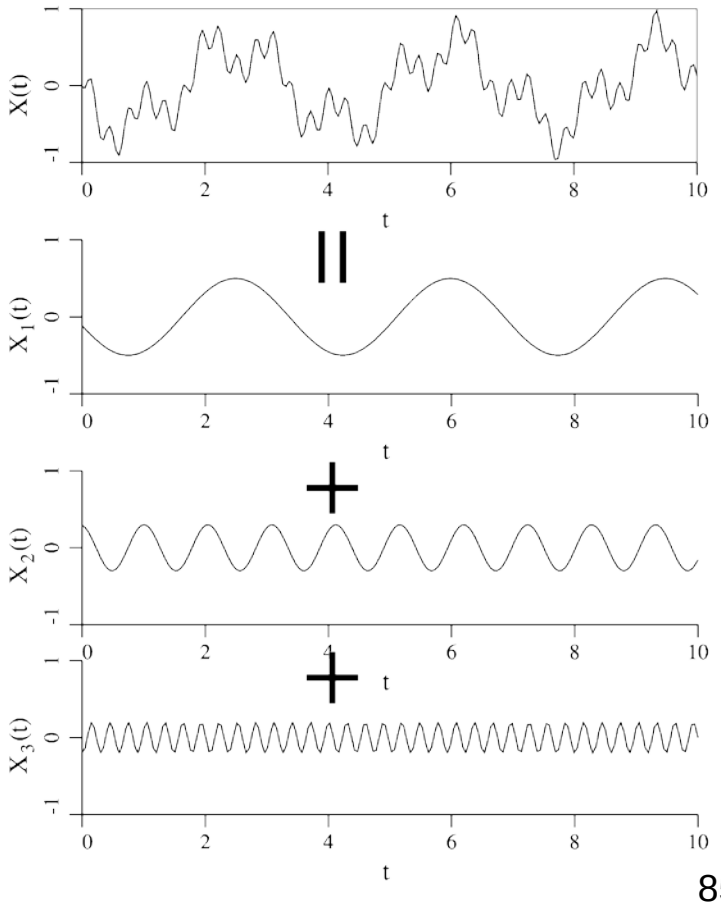
The computation of **L** becomes numerically unstable as *n* increases (e.g. a 200 x 200 field has *n* = 40000, so **L** and **C** are both 40000 x 40000 matrices)

Simulation of Random Fields

2) Fast Fourier Transform (FFT)

The random field is represented as a sum of sinusoids.

$$X(t) = C_1 \cos\left(\omega_1 t + \Phi_1\right) + C_2 \cos\left(\omega_2 t + \Phi_2\right) + C_3 \cos\left(\omega_3 t + \Phi_3\right)$$

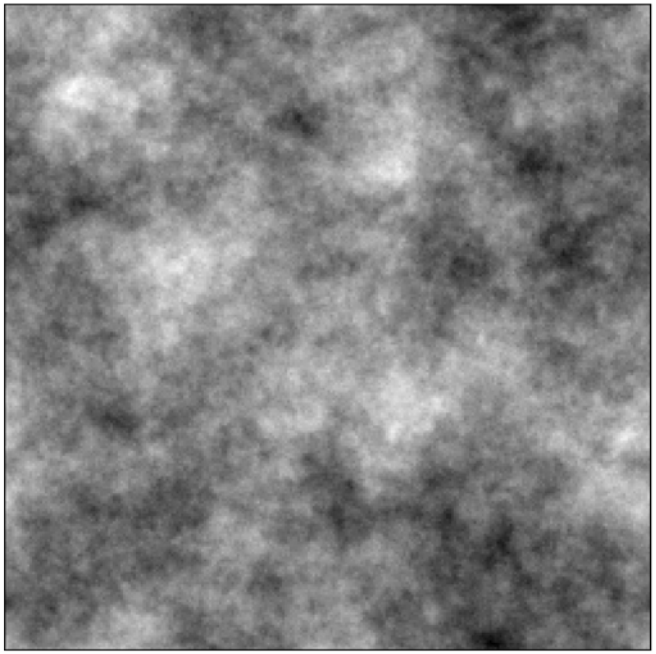


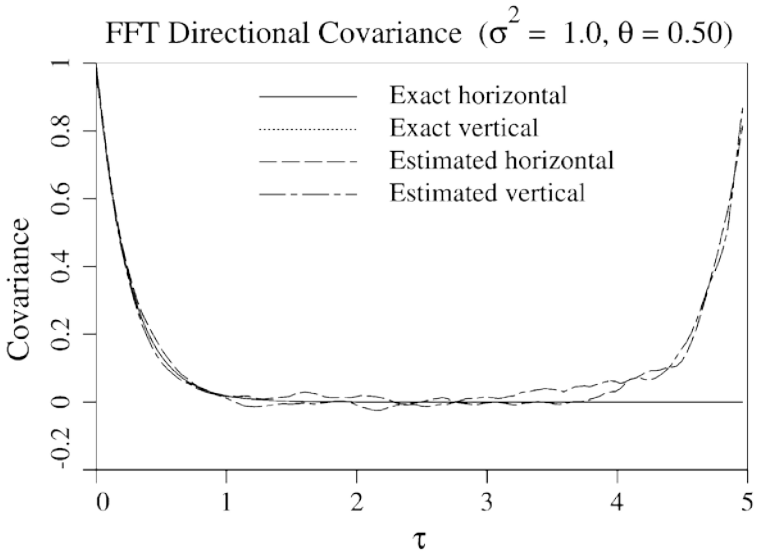
In general

$$X(t) = \sum_{k=1}^{N} C_k \cos(\omega_k t + \Phi_k)$$

$$= \sum_{k=1}^{N} \left\{ A_k \cos(\omega_k t) + B_k \sin(\omega_k t) \right\}$$

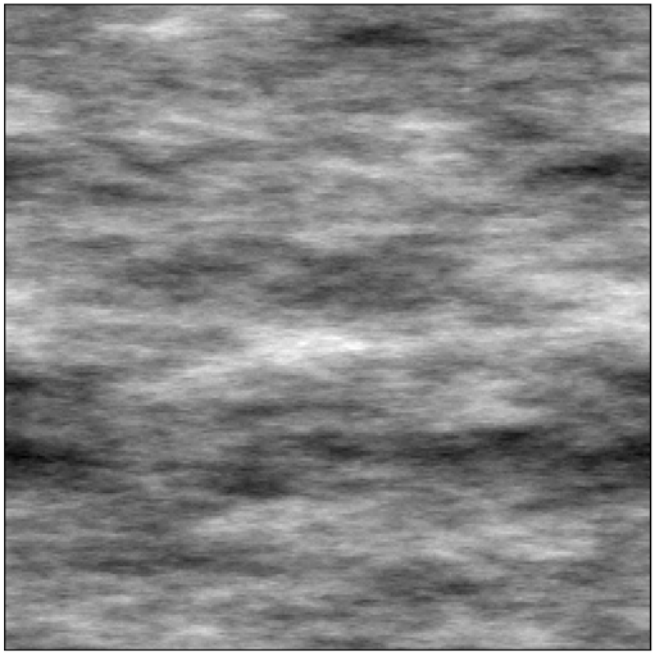
The FFT Simulation Method

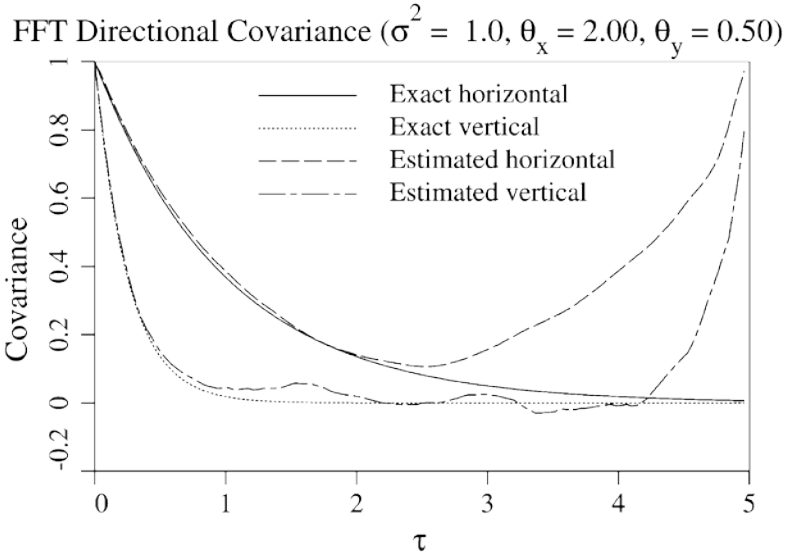




The FFT Simulation Method

Anisotropic Field:



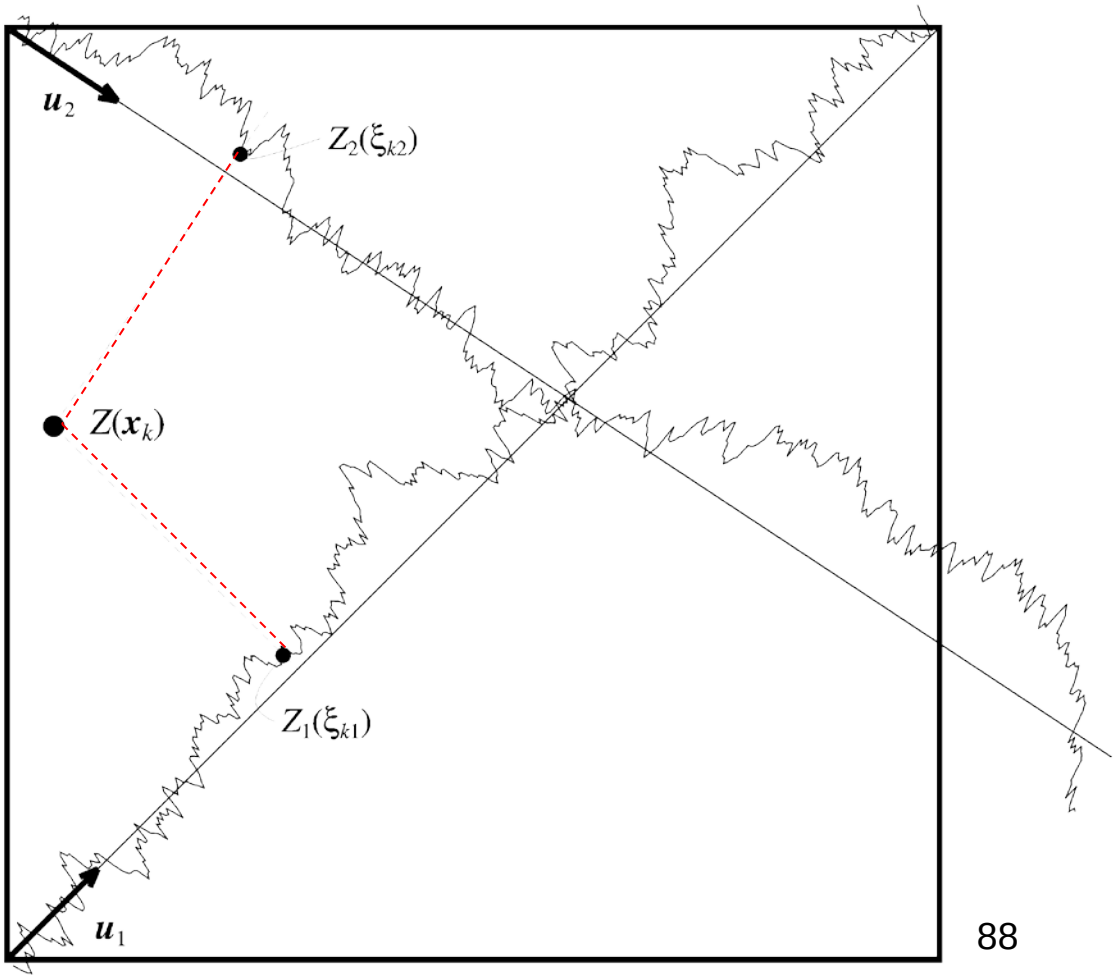


Simulation of Random Fields

3) Turning Bands Method (TBM)

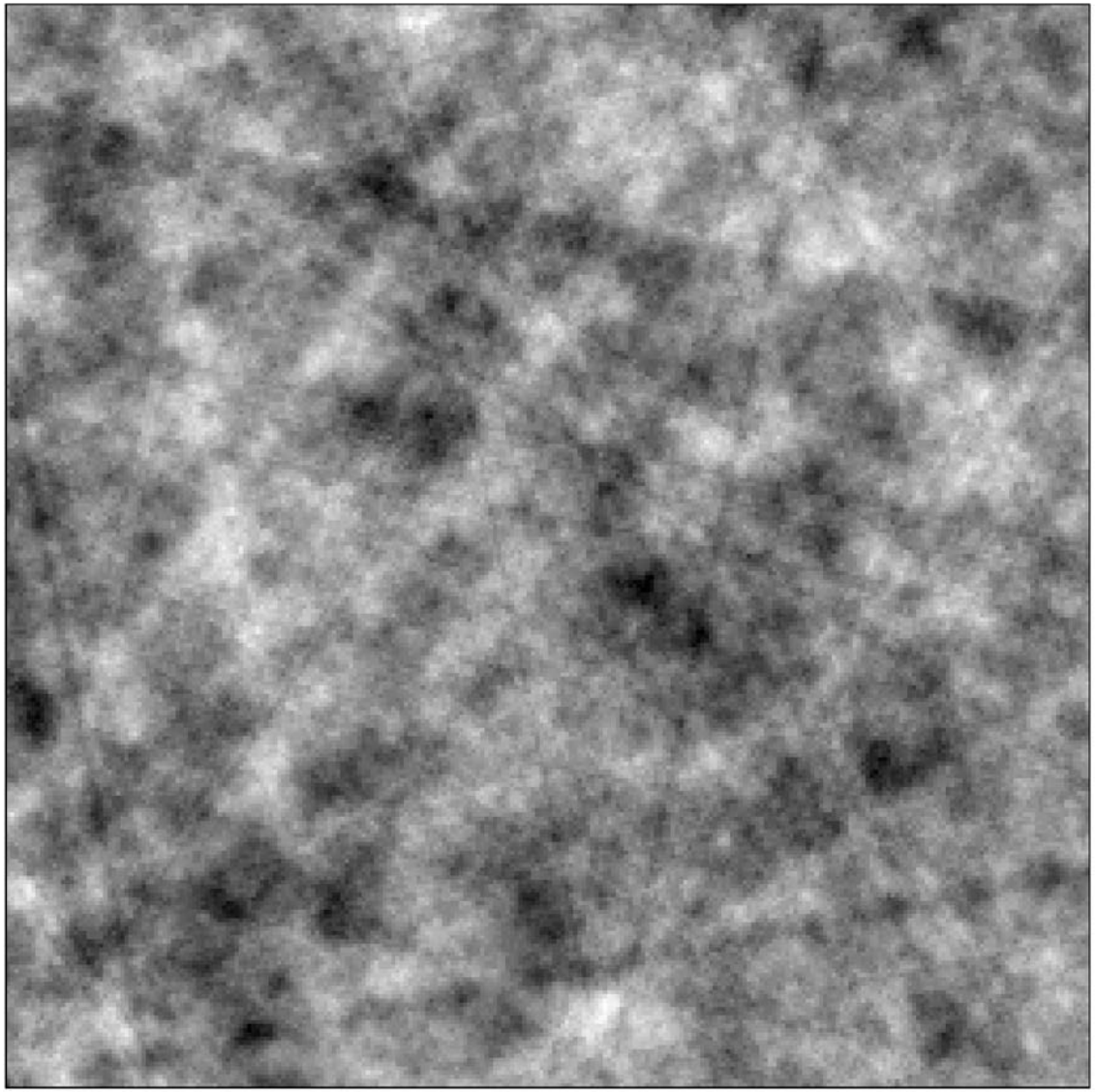
- 2 and 3-D fields constructed from a series of 1-D processes

$$Z(\underset{\sim}{x}_k) = \frac{1}{\sqrt{L}} \sum_{i=1}^{L} Z_i\left(\underset{\sim}{x}_k \cdot \underset{\sim}{u}_i\right)$$



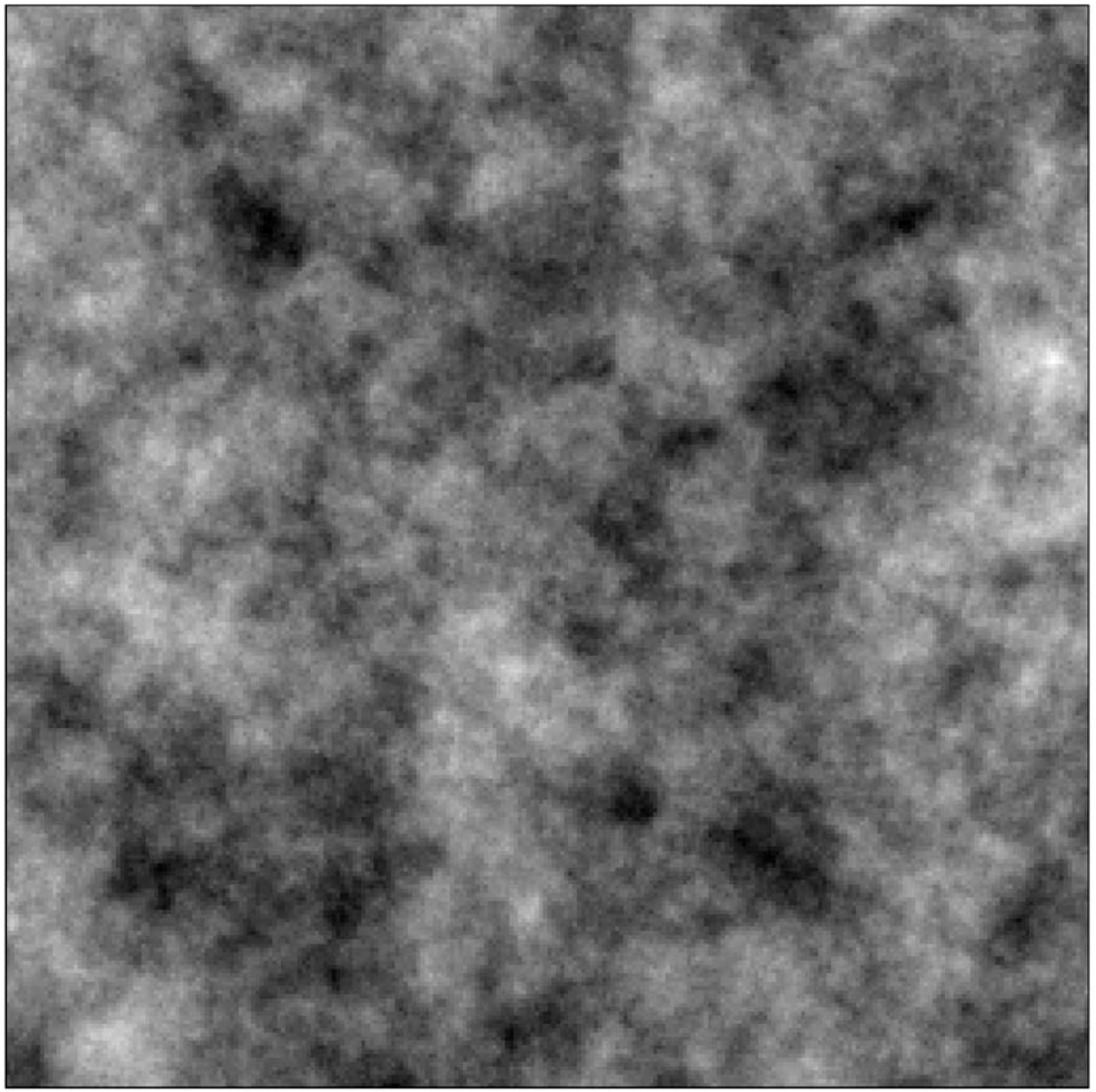
The Turning Bands Method

$L = 16$



The Turning Bands Method

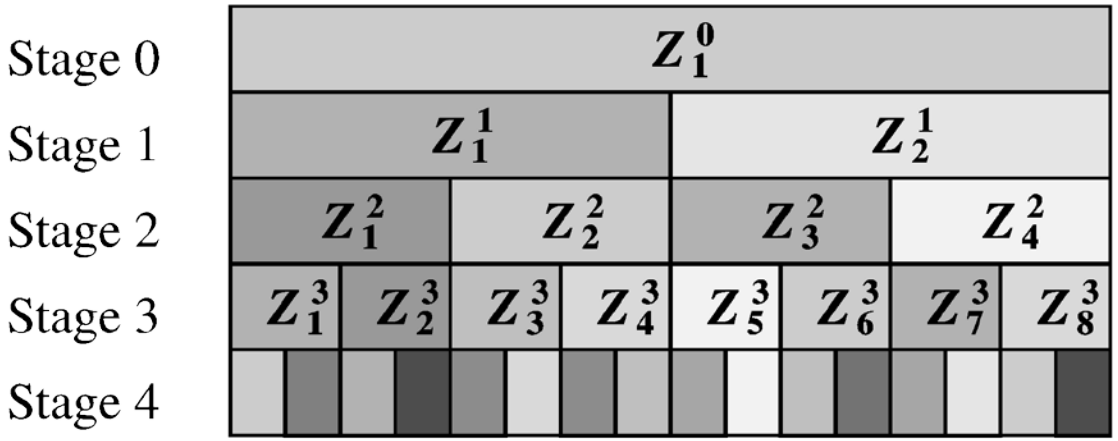
$L = 64$



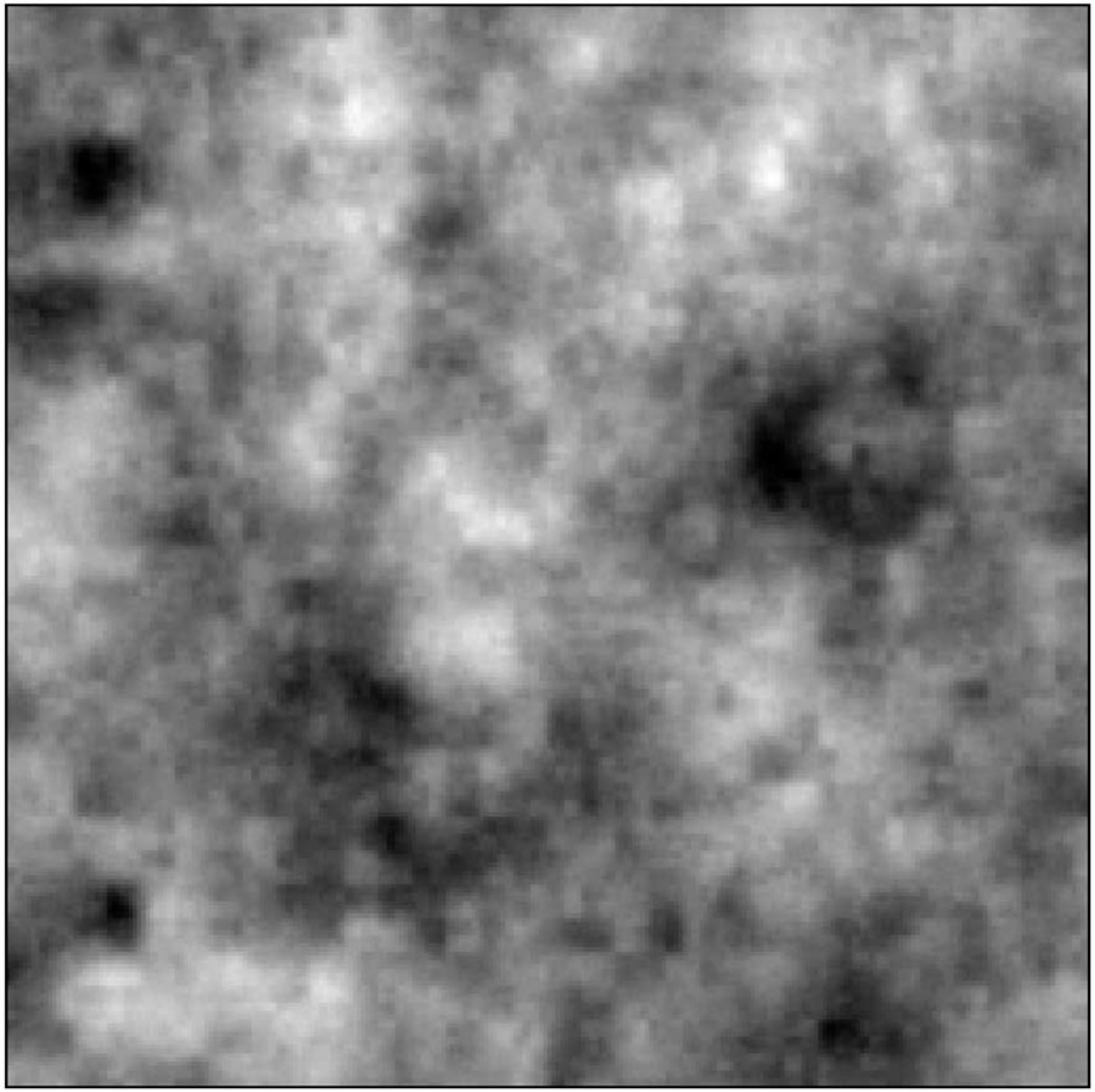
Simulation of Random Fields

4) Local Average Subdivision (LAS)

- produces local averages over a discrete set of cells
- appealing because almost all engineering properties are based on "local average" measurements (e.g. friction angle, cohesion, conductivity, etc.)
- statistics of the cell values correctly adjusted as the cell size changes
- top-down recursive algorithm is very efficient

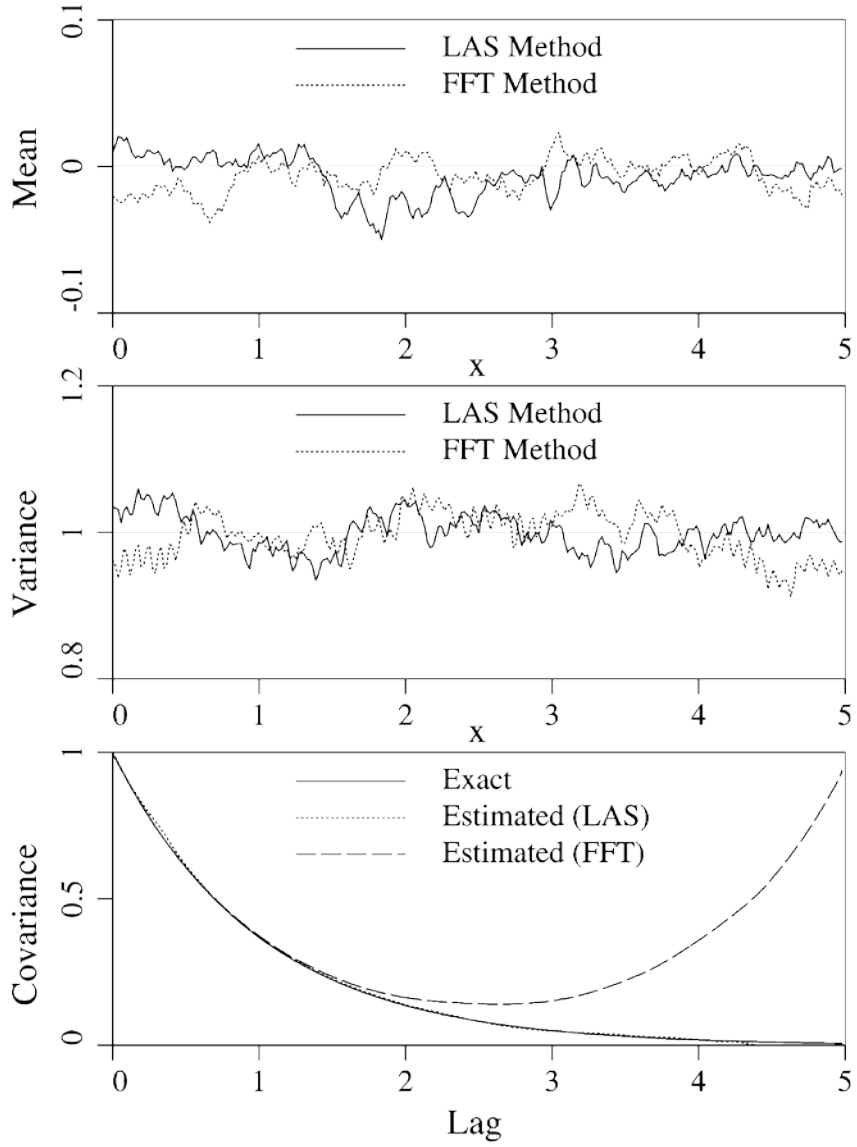


Local Average Subdivision

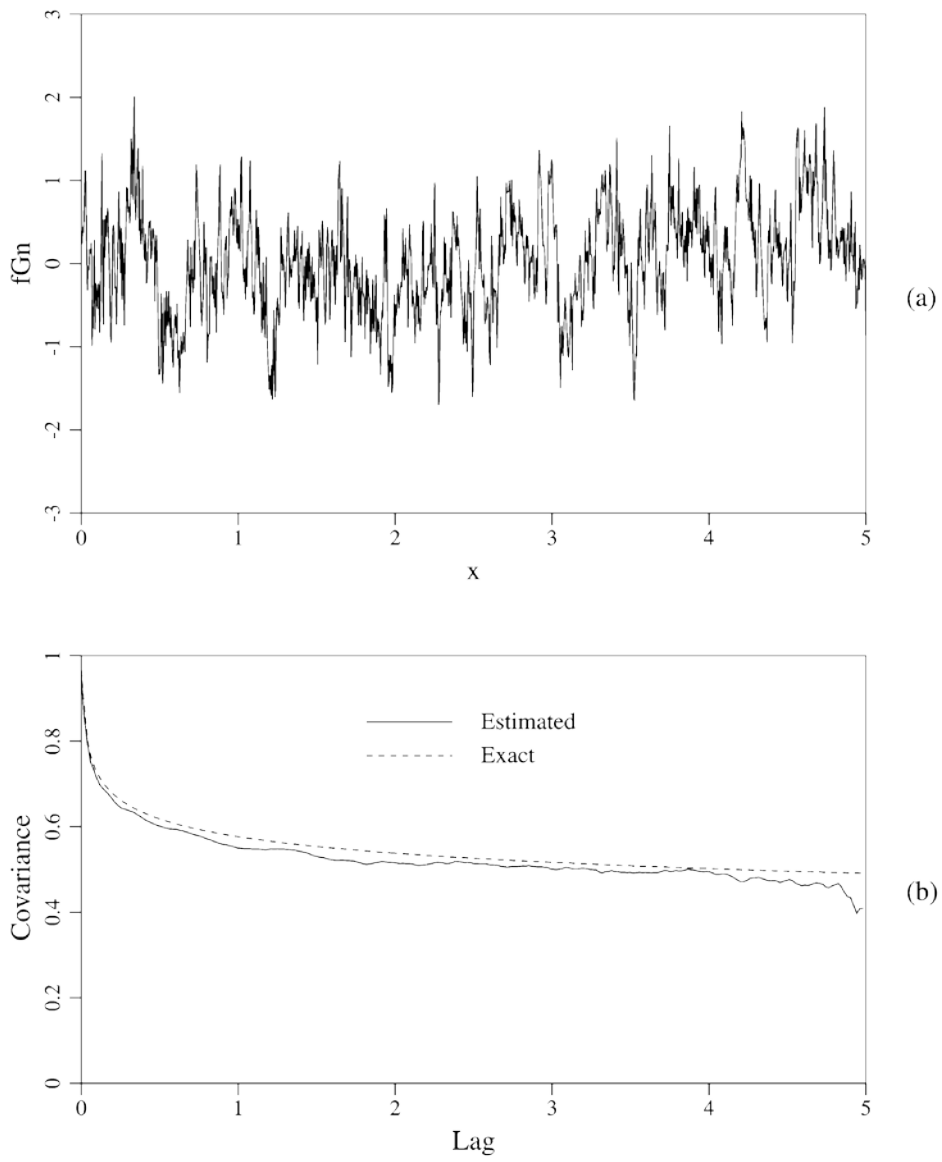


Local Average Subdivision

Comparison of LAS and FFT

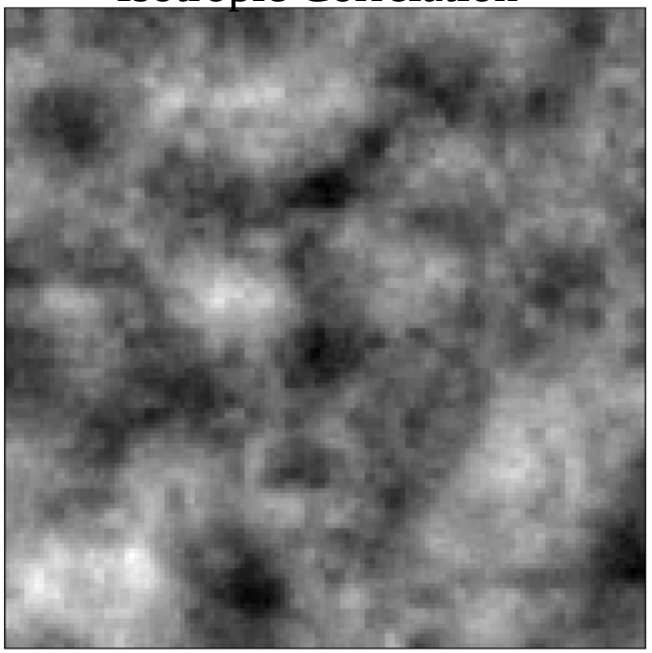


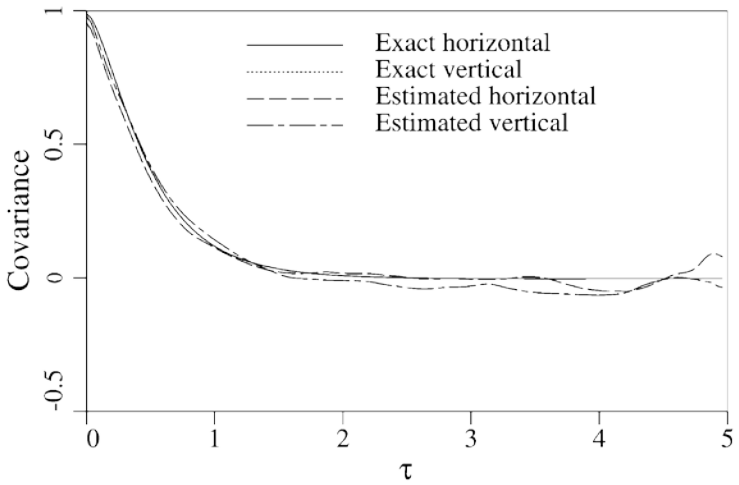
Local Average Subdivision



Local Average Subdivision

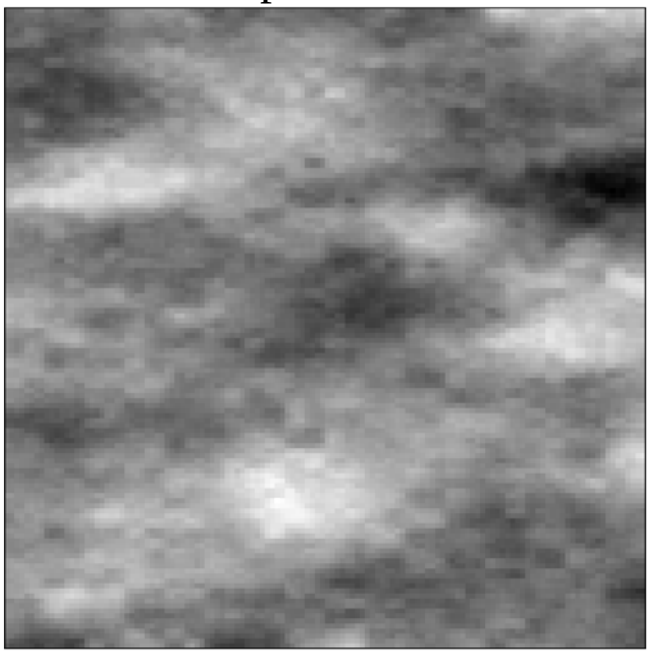
Isotropic Correlation





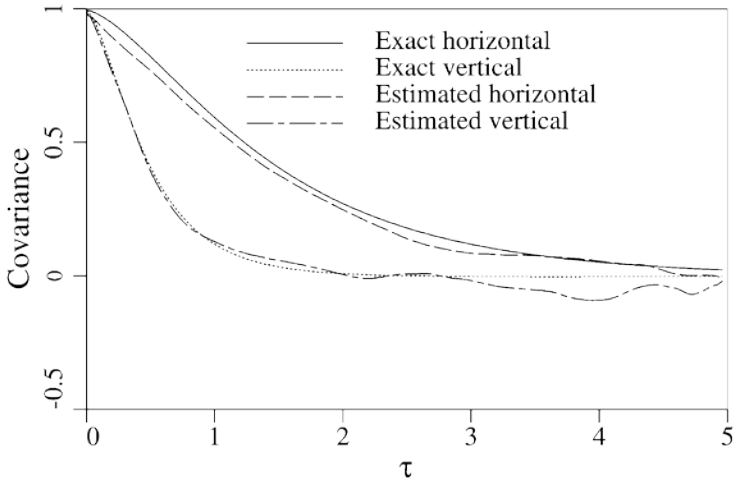
Local Average Subdivision

Anisotropic Correlation



$\theta_x = 3$

$\theta_y = 1$



Simulation of Random Fields

Summary of Approximate Methods

FFT

- reasonably efficient
- good for anisotropic processes
- need to know spectral density function
- covariance structure is symmetric about midpoint (to solve, generate fields twice as large as necessary)
- care must be taken with field discretization (to ensure frequency domain discretization is adequate)

Simulation of Random Fields

Summary of Approximate Methods

Turning Bands Method:

- reasonably efficient (depends on number of lines)
- accurate if sufficient number of lines are used
- not simply defined for anisotropic, nor non-standard, processes
- constructive/destructive interference leads to streaked appearance – difficult to quantify in higher dimensions.

Simulation of Random Fields

Summary of Approximate Methods

LAS

- very efficient
- easy to use (hard to code – boundary conditions are difficult to deal with)
- generates local averages – good for finite element modeling
- variance shows systematic pattern (of minor importance, although upon transformation to the lognormal distribution, the grid pattern shows up in the mean field – can be solved by shifting the field on each realization)

The Random Finite Element Method RFEM

The method involves a combination of
Random Field Theory (e.g. Fenton and Vanmarcke 1990)
with the
Finite Element Method (e.g. Smith and Griffiths 2004)

The method takes into account of the mean, standard deviation and spatial correlation length of the input parameters.

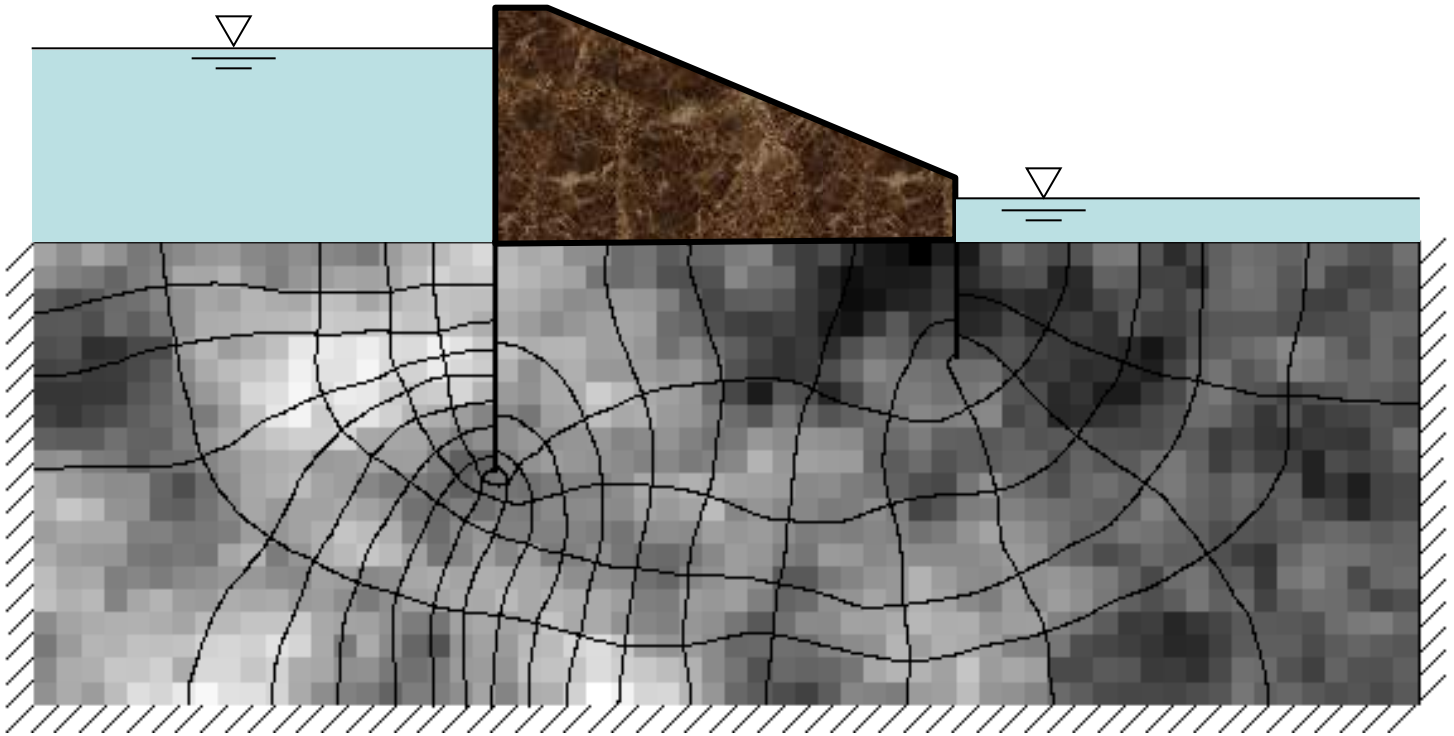
The method takes full account of local averaging of properties over the finite elements.

The method is applied in a Monte-Carlo framework.

Repeated calculations with the same input properties eventually lead to stable output statistics of the design parameters.

e.g. 2-d steady confined seepage (random permeability)

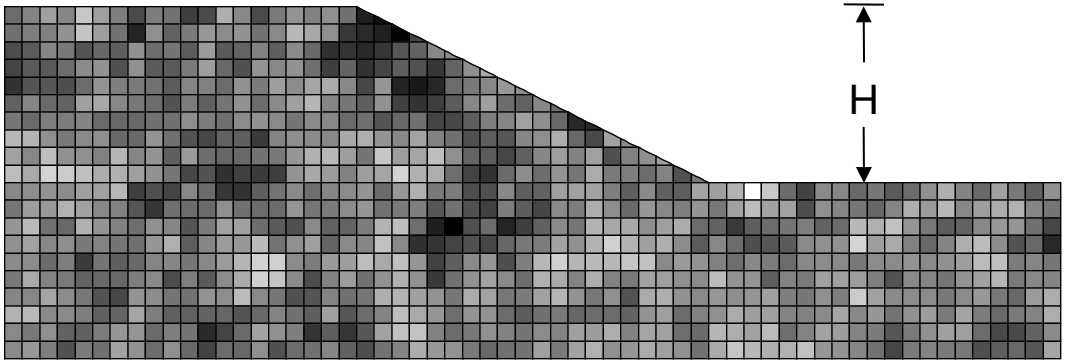
Dam with two cut-off walls



Dark zones indicate high permeability. Note how the equipotentials bunch together in the lighter zones where the permeability is low

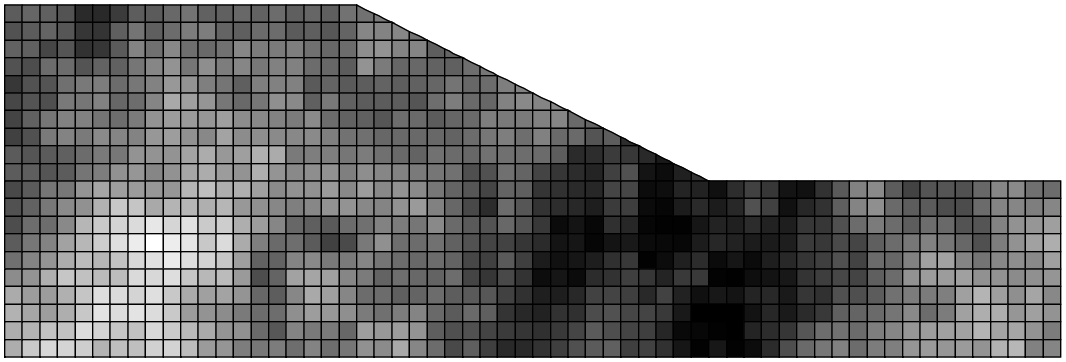
RFEM IN SLOPE STABILITY ANALYSIS

The correlation length recognises that soil samples “close” together in the field are more likely to have similar properties that if they are “far apart”



Low spatial correlation

$$\theta = 0.2H$$

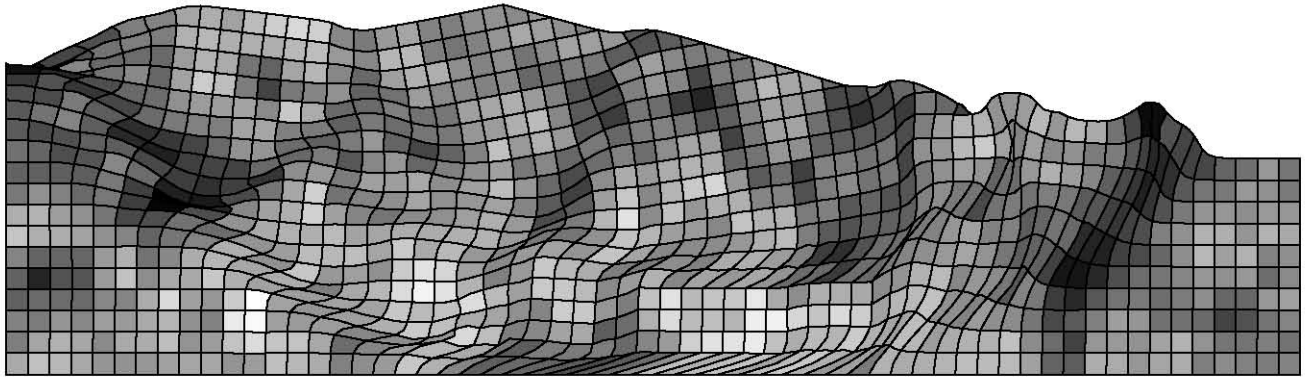


High spatial correlation

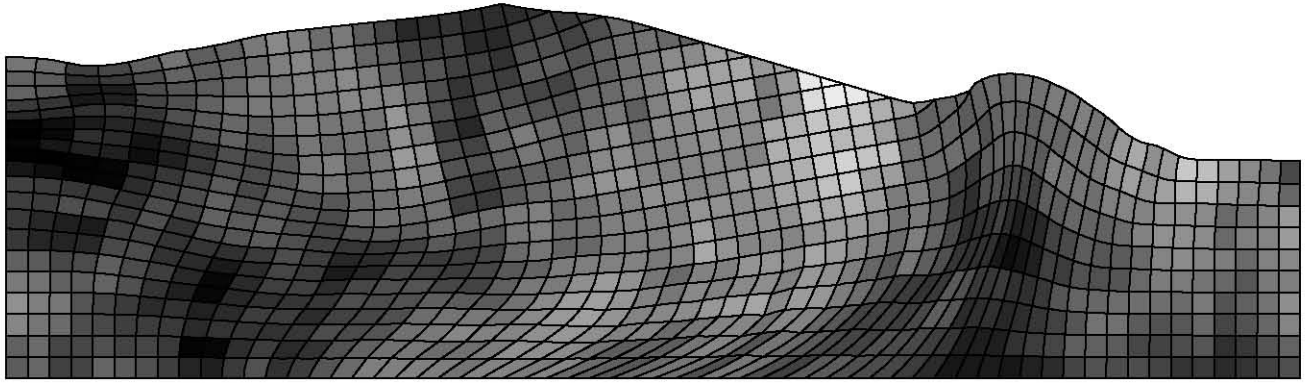
$$\theta = 2H$$

Both slopes may have the same mean and standard deviation of shear strength, But quite different spatial correlation lengths.

$\Theta_C = 0.5$



$\Theta_C = 2$



Typical random field realizations and deformed mesh at slope failure for two different spatial correlation lengths.

RFEM IN BEARING CAPACITY ANALYSIS

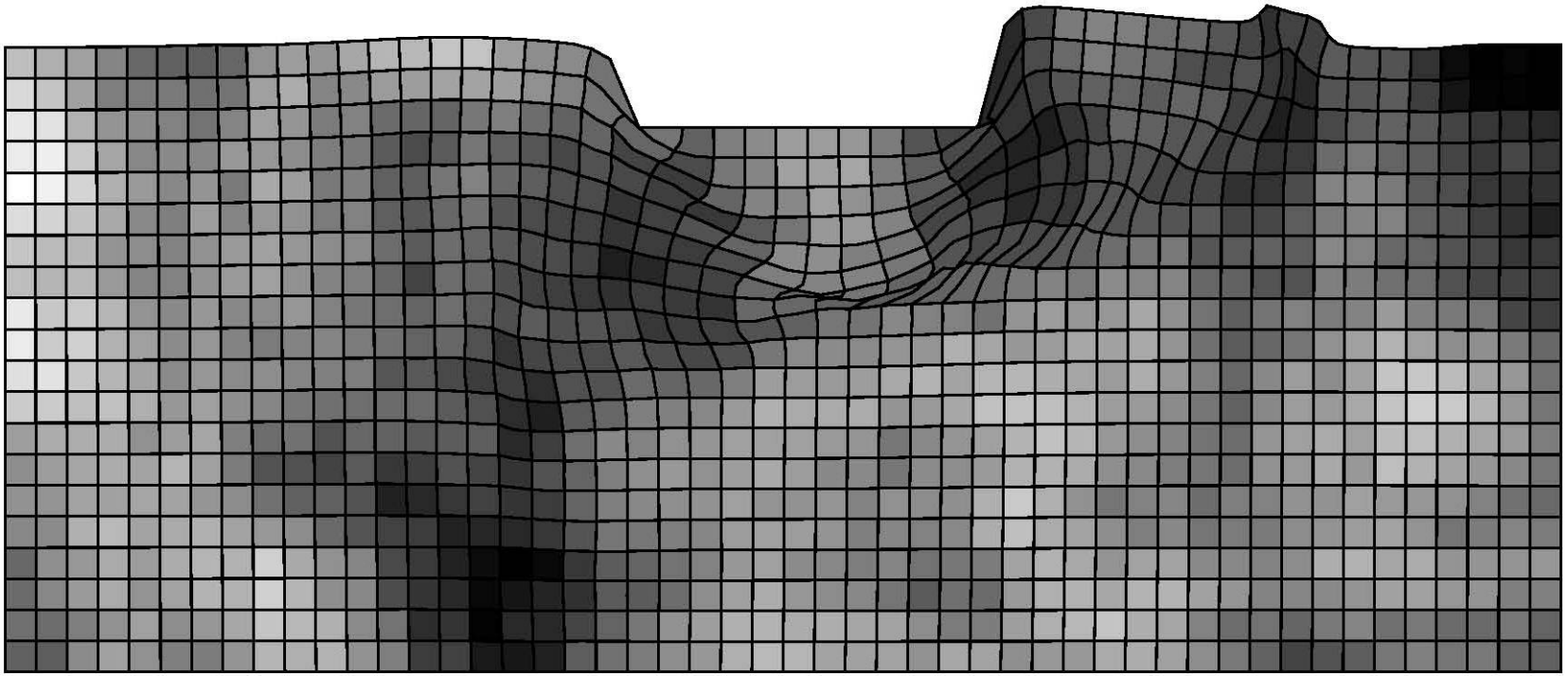
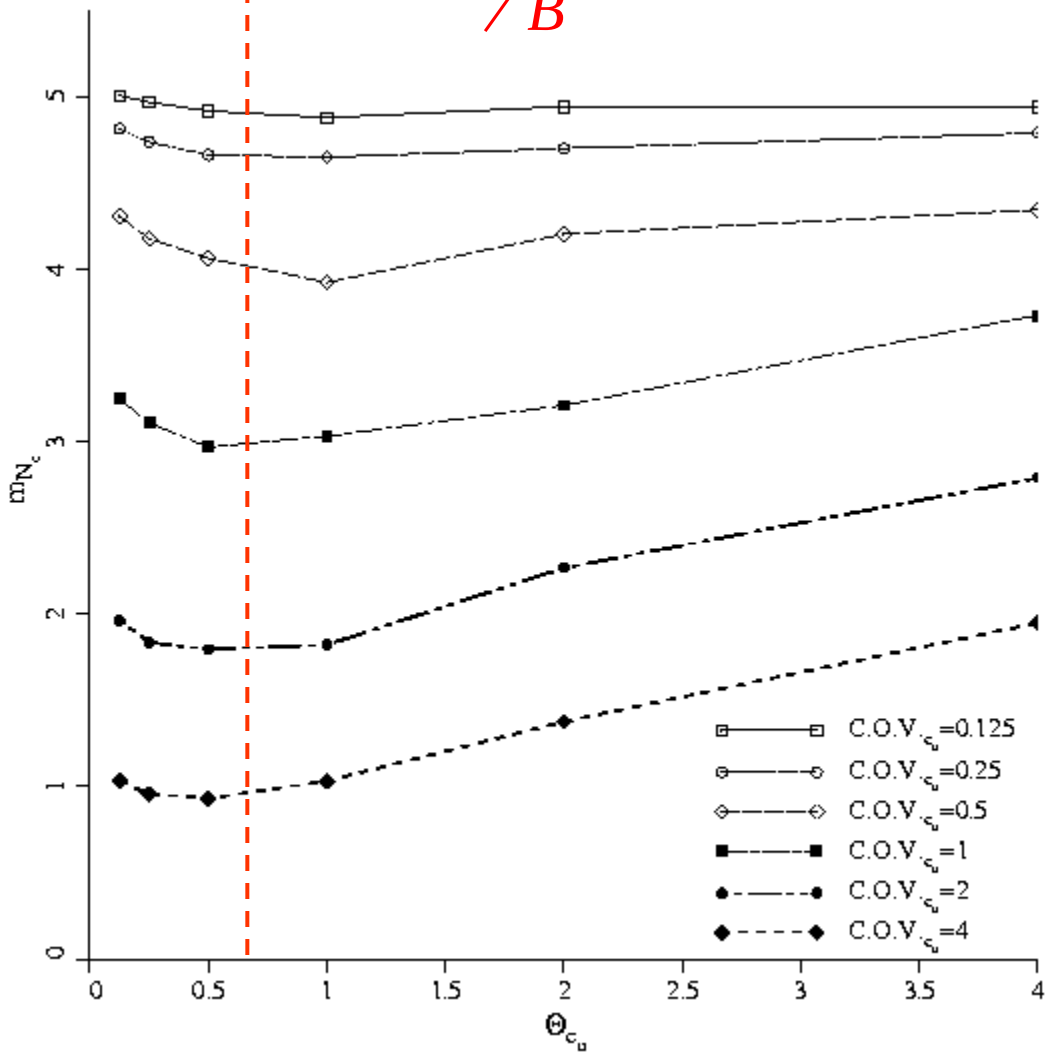


Fig 4. Typical deformed mesh and greyscale at failure with $\theta_{1nc_u}/B=1$. The darker regions indicate weaker soil

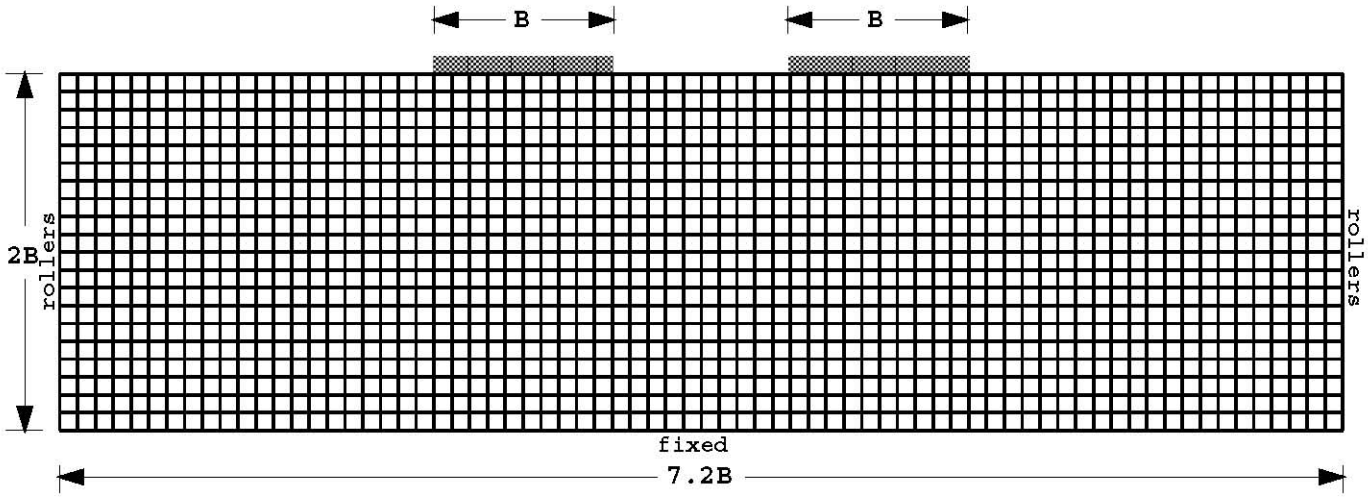
"worst case" correlation length

$0.5 < \theta/B < 1$

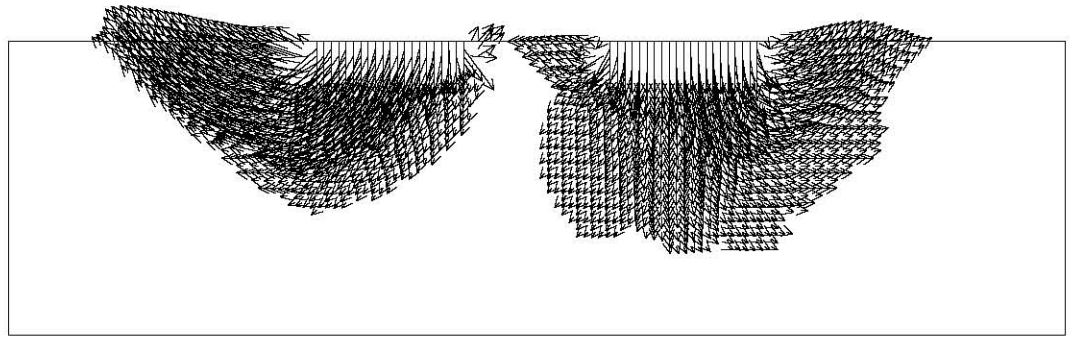


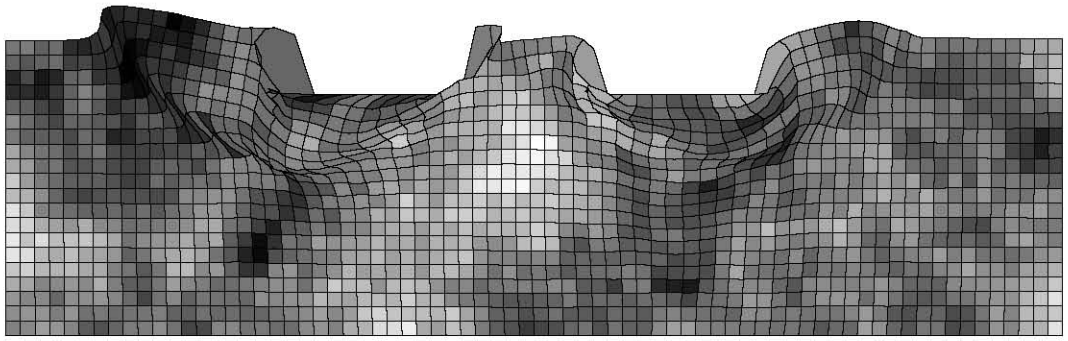
Estimated mean bearing capacity factor m_{N_c} as a function of Θ_{c_u} and COV_{c_u}

RFEM IN BEARING CAPACITY ANALYSIS



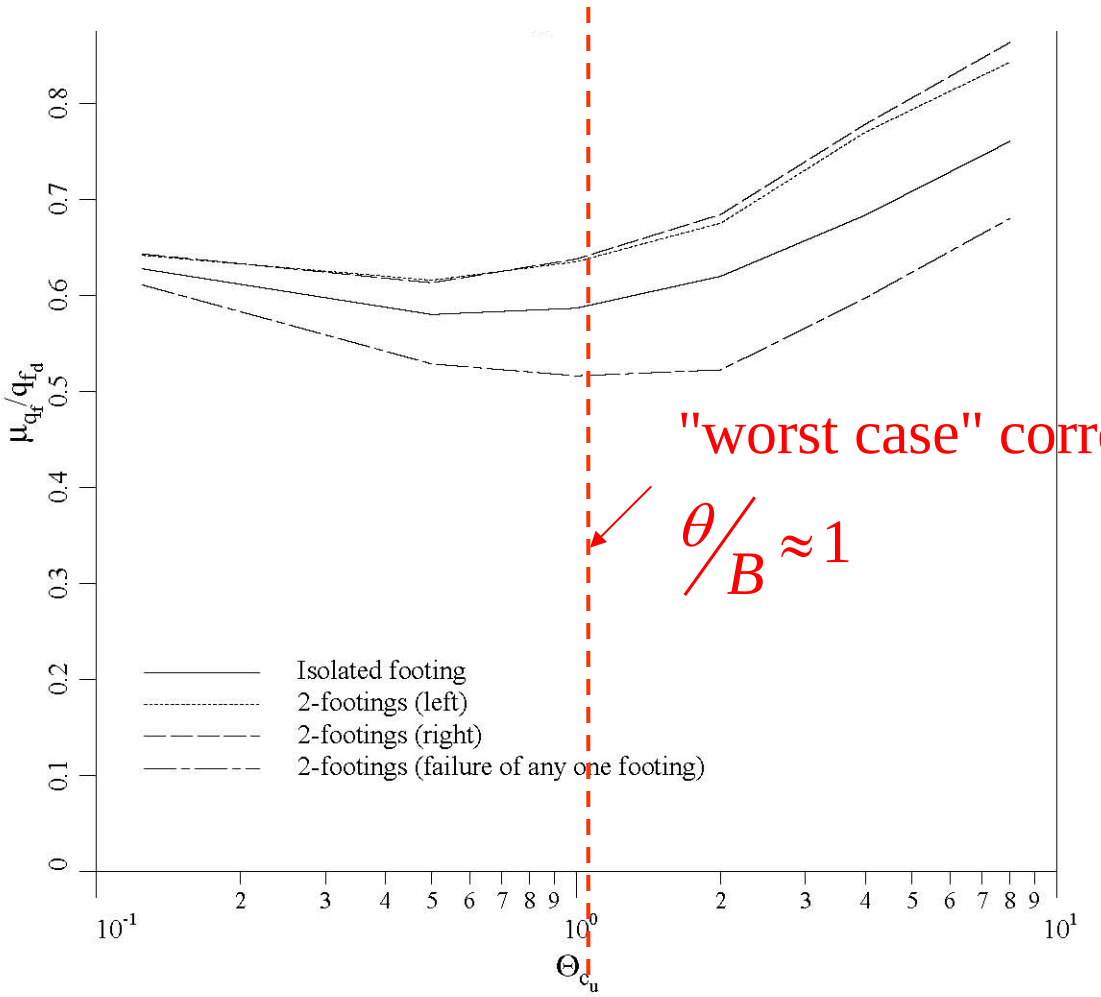
b) Two footings





a)

Typical displacement vectors and deformed meshes at bearing failure in a two-footing analysis on a stochastic soil.



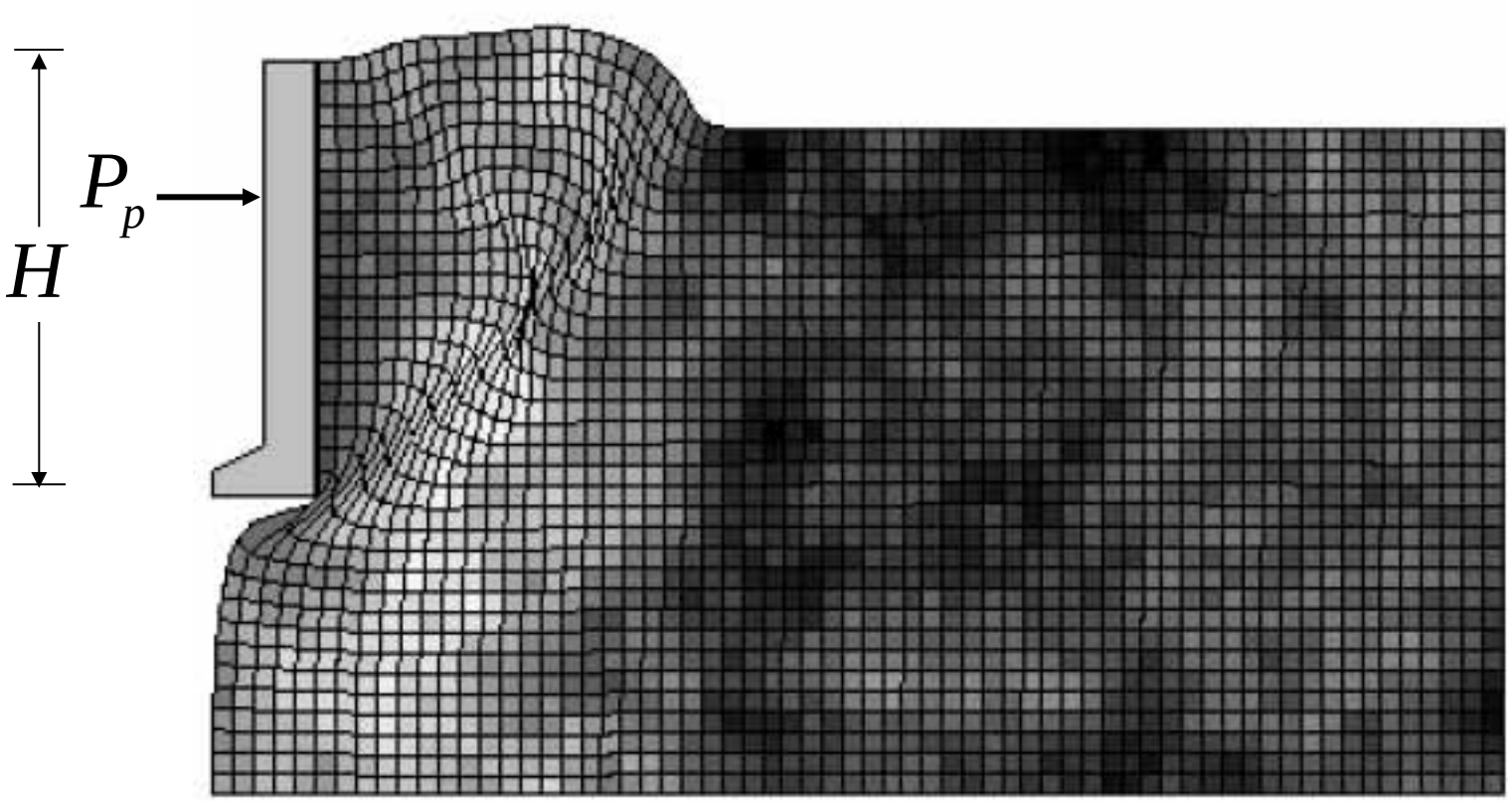
"worst case" correlation length

$\theta/B \approx 1$

Influence of Θ_{c_u} on normalized mean bearing capacity for COV_{c_u}=1.0

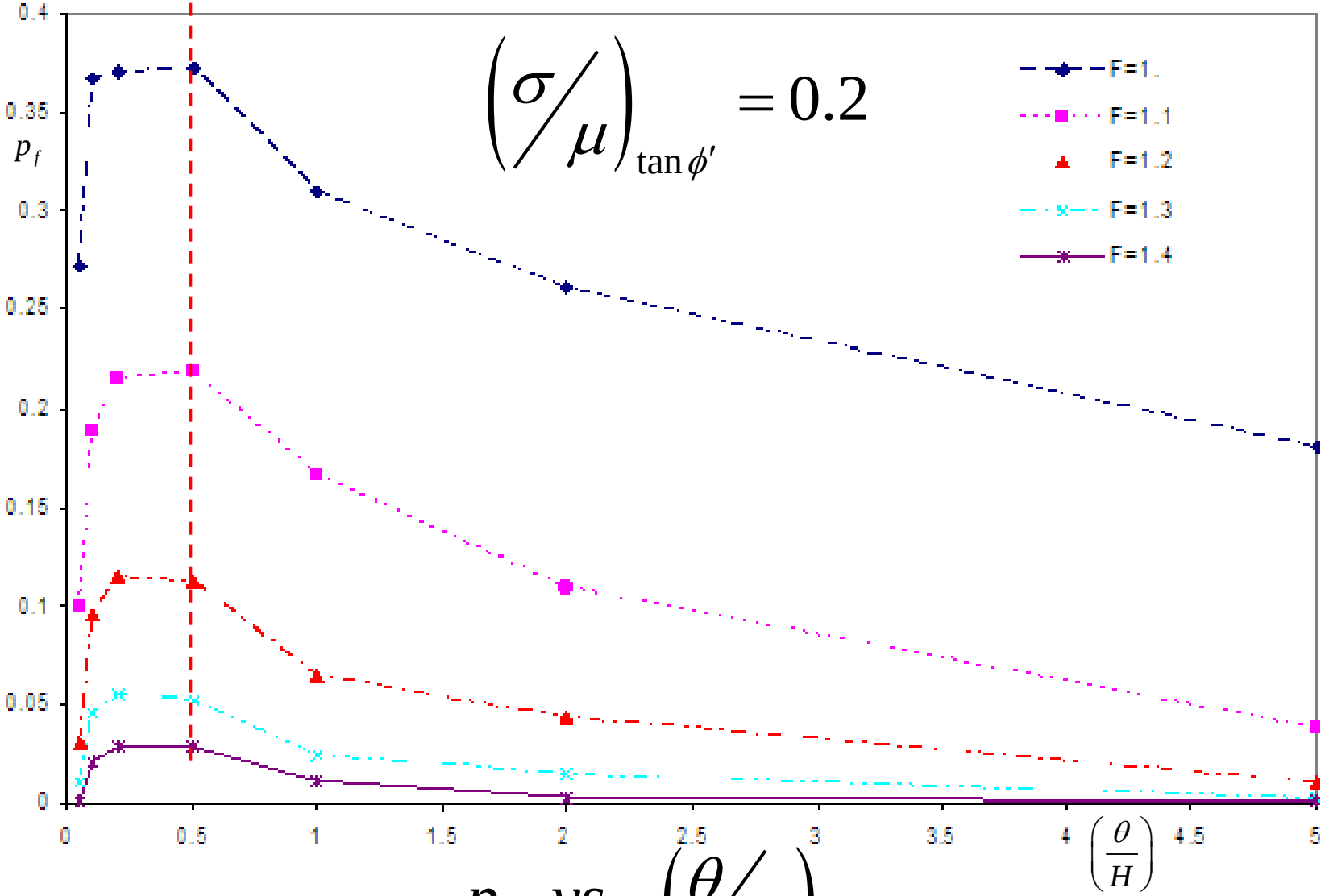
RFEM IN EARTH PRESSURE ANALYSIS

$$\frac{\theta}{H} = 1$$



"worst case" correlation length

$\theta/H \approx 0.5$



p_f vs. $\left(\theta/H\right)$

Concluding Remarks

The Random Finite Element Method (RFEM) offers many advantages over conventional probabilistic analysis tool—especially for nonlinear analysis.

-no *a priori* judgment relating to the shape or location of the failure surface. The FE analysis "seeks out" the critical mechanism.

-a "worst case" spatial correlation has been clearly identified leading to the highest probability of failure. This has important implications for design.

-probabilistic tools that essentially analyze "homogenized" problems, albeit with corrected properties that account for spatial correlation via local averaging are unable to detect this effect

-the RFEM results show that simplistic probabilistic tools may be unconservative for correlation lengths close to the "worst case" value.

The Random Finite Element Programs described above are already available for free download as part of the textbook:

"Risk Assessment in Geotechnical Engineering"
by
Gordon A. Fenton and D.V. Griffiths
Wiley (2008)

rbear2d	**Bearing capacity (2d)**
rdam2d	**Freesurface flow (2d)**
rearth2d	**Limiting earth pressure (2d)**
rflow2d	**Steady seepage (2d)**
rflow3d	**Steady seepage (3d)**
rpill2d	**Mine pillar stability (2d)**
rpill3d	**Mine pillar stability (3d)**
rsetl2d	**Footing settlement (2d)**
rsetl3d	**Footing settlement (3d)**
rslope2d	**Slope stability (2d)**

www.engmath.dal.ca/rfem