3

Data Science for Uncertainty Quantification

3.1. INTRODUCTORY EXAMPLE

3.1.1. Description

This chapter provides an overview of the relevant mathematical, statistical, and computer science components needed to develop and understand the various approaches to uncertainty quantification in subsequent chapters. The aim here is to place these various components within the unified context of uncertainty quantification, and as application to geoscientific fields and not just to provide a mere overview, which can be found in many excellent existing books. Instead, we provide some additional insight into how various, seemingly independent applied mathematical concepts share important common traits within the context of uncertainty quantification. The field of data science (statistics, machine learning, and computer vision) is growing fast. Most developments are driven by a need to quantify human behavior or interaction (e.g., Facebook). Here we deal with the physical world. Data scientific approaches will need to be tuned to the kind of challenges we face in this world, such as data sparsity, complex relationship between physical variables, high dimension, non-Gaussianity, nonlinearity, and so on.

As an aid to this overview, we develop a simple illustrative example. This example is not a real example by any stretch of the imagination but has elements common to uncertainty quantification as outlined by the real field studies in the previous chapter. This will also aid in developing notation in this book by providing common notation in the various applied mathematical and computer science disciplines involved. The reader should also refer to the notation in the next two sections.

The case here concerns the simple design of a water purification system by pumping and infiltrating river water into an aquifer by means of an infiltration basin (see Figure 3.1). Then, usable drinking water is retrieved from a pumping well. The aims of this artificial recharge is

1. to improve groundwater quality through filtration and bioactivity in the soil
2. to create a hydraulic barrier to divert any contaminated groundwater from an known industrial polluting area

However, the pumping wells needs to be shut off from time to time, either for saving energy or for maintenance. Too long of a shut-off period may result in pollutants infiltrating the filtration zone. Hence, knowing whether such contamination will take place and knowing when this will take place will help in designing shut-off periods. To aid in this design, head measurements from several wells in the area are available, as well as reports that the subsurface is substantially heterogeneous because of the depositional system induced by the nearby river (fluvial system).

3.1.2. Our Notation Convention

One of the challenges in scientific writing is to come up with a strategy for representing common objects, vectors, scalars, function, random functions, and so on. Many publications and books use different conventions and notations. Our material comes from different worlds with different notations, so we do not want to leave the reader guessing what "x," or "i," or "n" is; hence, we are quite explicit on notation in the hope this also unifies many concepts. Here is an overview what you need to know:

1. a, b, x: small italic is a scalar, an outcome, a sample
2. A, B, X: capital italic is either a matrix or a random variable. Those different contexts are usually clear.
3. $\mathbf{X}$: capital bold is a random vector (a vector of random variables)
4. $\mathbf{x}$: small bold is the outcome of a random vector or simply a vector
5. L: the number of samples in a statistical study
6. N: the dimension of the problem in general, for example, the dimension $\mathbf{X}$

Quantifying Uncertainty in Subsurface Systems, *Geophysical Monograph 236*,
First Edition. Céline Scheidt, Lewis Li, and Jef Caers.
© 2018 American Geophysical Union. Published 2018 by John Wiley & Sons, Inc.

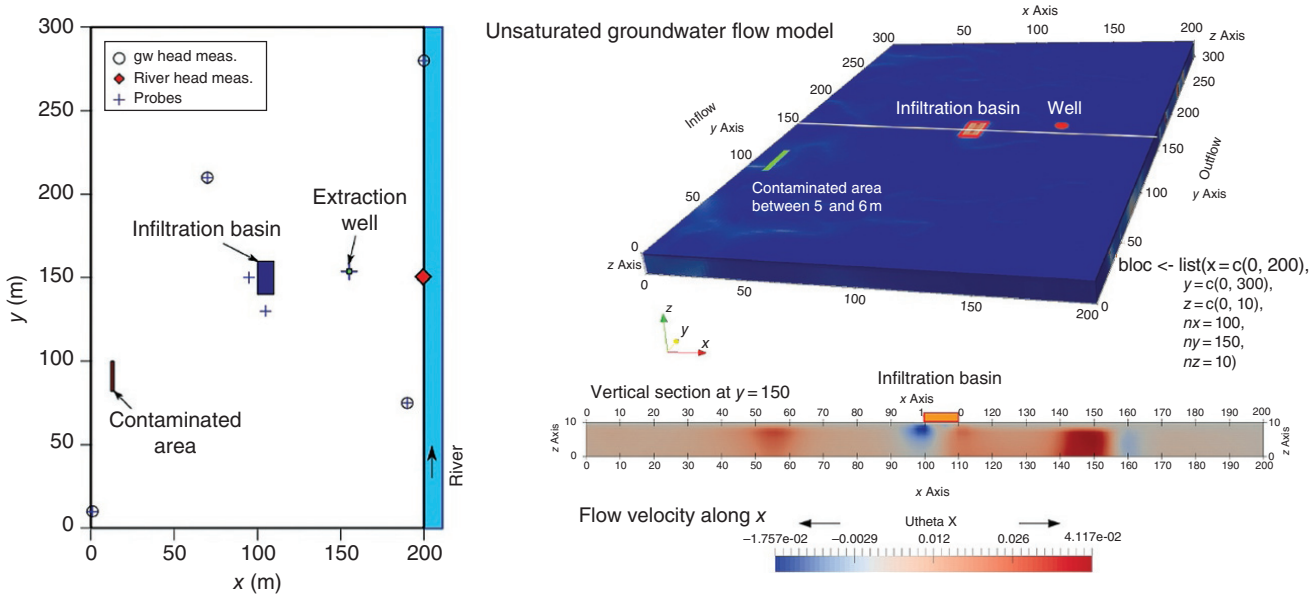


Figure 3.1 Setup of our simple illustration case.

7. *Counters*: we use small font and corresponding capital letter, for example, $n = 1, \ldots, N$ or $\ell = 1, \ldots, L$. We avoid using mixing letters such as $i = 1, \ldots, N$
8. f: probability density function (pdf) or just a function
9. F: cumulative distribution function (cdf)

3.1.3. Variables

Since we are interested in the future evolution of a system, we need to model and simulate the above-presented situation with computer models. Basically, any UQ exercise has a number of components as synthetized in Section 1.7. For any UQ problem, it is therefore critically important to clearly and rigorously define the data variables, the model variables, and the prediction variables. A common confusion is to think that the data variables are the observed data. They are not. In a typical probabilistic way of reasoning, the actual field data are seen as one particular realization, sample, or instantiation (whatever choice of nomenclature one is used to) of these data variables. Data variables are random variables whose outcomes are not known. How do we define such data variables? Data is caused by some action (a sampling, a measurement, a study, a drilling, a logging, etc.) on the subsurface system in question. Hence, in order to define data variables, we need to first define the variables that describe the (unknown) subsurface system. These are the model variables. These model variables are, therefore, the parameterization that the modeler believes to allow for a proper representation of all aspects of the subsurface system, whether these are fluxes, pressures, concentrations, chemical reactions kinetics, and so on.

In the hydro case mentioned earlier, we will represent the area in question with a number of grid cells. Many simulators, for example of multiphase flow, of reactive transport, and of geostatistical algorithms, require a grid and a definition of the size of these cells. These cells may be on a regular grid or on any grid, depending on how accurately models need to be simulated. Here we assume the grid is regular and has a certain cell size. To model this system, we will need to specify spatially distributed values such as porosity and hydraulic conductivity. A subset of the model variables are the relevant properties for each cell value. However, property values in different grid cells are not independent (statistically). The geological depositional system has induced spatial correlation and such spatial correlation is often modeled using geostatistical models or algorithms. As such, the gridded model variables depend on the definition of some statistical model, which may have its own parameter/model variables, for example the mean porosity or in the case of a correlated spatial porosity, the spatial covariance or variogram model parameters. In addition, the above model requires defining initial conditions and boundary conditions, both of which may be uncertain and also modeled using probability distributions with their own set of parameters.

In this book, because of the specific spatial (or spatio-temporal) nature of subsurface models, we split the model variables in two groups: (i) those model variables that comprise the spatial distribution of properties on a grid, for example concentration, porosity, and permeability

at each grid location and (ii) the parameters that were used to generate these spatial model variables, or any other model variables or parameters that are not defined on the grid. For the model parameterization, as a whole, we use the notation **m** comprising of (i) gridded model variables and (ii) non-gridded model variables.

$$\mathbf{m} = \left(\mathbf{m}_{\text{grid}}, \mathbf{p}\right) \tag{3.1}$$

Another part of the model that will be dealt with separately is the physical/chemical/biological process that is modeled using ODEs or PDEs or whatever other mathematical representation one deems appropriate. Such equations can be seen as "theories" (in the mold of *Tarantola* [1987]) that provide information on the relationship between several aspects of the subsurface system, for example stating the theoretical link between data and model variables. In this book, we will limit ourselves to expressing these relationships using forward models. A forward model is represented by an explicit function (derived from an implicit physical relationship). A first forward model is between the data variables and the model variables

$$f_d(\mathbf{m}, \mathbf{d}) = 0 \Rightarrow \mathbf{d} = g_d(\mathbf{m}) = g_d\left(\mathbf{m}_{\text{grid}}, \mathbf{p}\right) \tag{3.2}$$

For example, f_d could be a set of partial differential equations and g_d the numerical implementation that outputs the desired data variables. Therefore, g_d represents the data forward model, a simulation model that takes as input the model variables and outputs the data variables. For the hydro case in question, we can now specify both model variables and data variables. As model variables, we have the following (summarized in Table 3.1):

1. *Hydraulic conductivity* (a gridded property). The parameters used to generate a gridded hydraulic conductivity model are uncertain. Here we assume that hydraulic conductivity can be modeled using a Gaussian process (see Section 3.7.5). Such process requires specifying the mean, standard deviation, and a set of spatial covariance parameters such as range, nugget, and anisotropy, as well as type of covariance model.
2. *Boundary conditions*. Boundary conditions are simulated by means of Gaussian process regression. The Gaussian process regression is here defined by a prior mean and Matérn covariance function. This covariance is not known, and its uncertainty is parametrized as uncertainty in the variance, range, and smoothness. The prior mean function is specified by three monomial basis functions with unknown parameters with a vague prior. The Gaussian process is conditioned to four groundwater

Table 3.1 Overview of the various parameters and their uncertainty for hydro case.

	Parameter code	Description	Variable type	Distribution
Hydraulic conductivity representation	K_{mean}	Mean value of hydraulic conductivity K (m/s)	Continuous	U(7e–4, 10^{-3})
	K_{sd}	Standard deviation of log(K) (m/s)	Continuous	U(0.05, 0.3)
	K_{Cov}	Type of covariance model for simulation of K	Discrete	Gaussian or spherical
	K_{angle}	Horizontal anisotropy angle for K (degree)	Continuous	U(110, 150)
	K_{range}	Correlation length along the principle direction for K (m)	Continuous	U(10, 100)
	$K_{\text{anixy\_ratio}}$	Anisotropy, horizontal stretching ratio	Continuous	U(l/20, 1/2)
	$K_{\text{aniz\_ratio}}$	Anisotropy, vertical stretching ratio	Continuous	U(15, 30)
	K_{nugget}	Nugget for K (m/s)	Continuous	U(0, 0.1)
Boundary conditions representation	H_{sd}	STD of the Matern covariance model for simulation of boundary conditions	Continuous	U(0.01, 0.1)
	H_{range}	Correlation length of the Matern covariance model for simulation of boundary condition	Continuous	U(20, 40)
	H_{nu}	Smoothness of the Matern covariance model for simulation of boundary conditions	Continuous	U(1.5, 3.5)
	H_{rivGrad}	Gradient of the river	Continuous	N(–0.0015, 0.0001)
Measurement error	H_{rivRef}	River hydraulic head (meters)	Continuous	N(7, 0.05)
	H_{nugget}	Measurement error groundwater hydraulic heads (meters)	Continuous	U(0.02, 0.1)

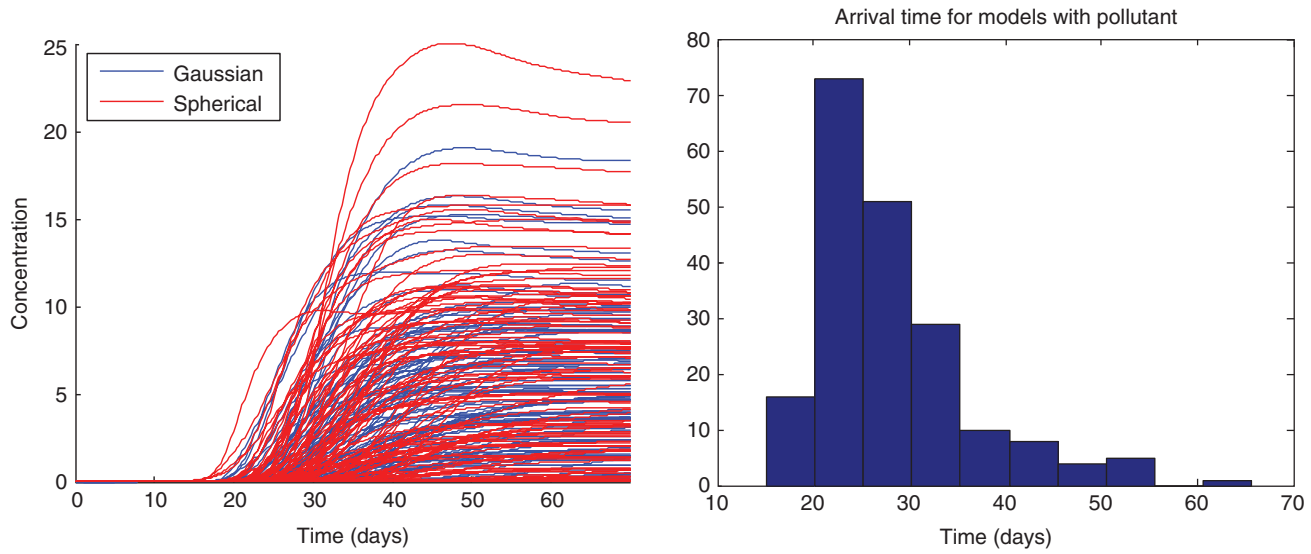


Figure 3.2 Concentration of DNAPL at the drinking well location with histogram of arrival times.

hydraulic head measurements as well as to the river heads (with measurement error, see the following text). The river head gradient is uncertain as well.

3. *Measurement error*. The hydraulic head measurements are subject to noise, modeled as a noise variance (entering as a nugget effect when estimating the boundary conditions). The hydraulic head of the river is uncertain as well.

Essentially, the model is infinite dimensional, unless one discretizes space and time. Because both space (grid cells) and time (time-steps) are usually discretized, the model is very high dimensional, namely assuming all model variables are time-varying

$$\dim(\mathbf{m}) = N_{\text{timesteps}}\left(N_{\text{gridcells}} \times N_{\text{properties}} + \dim(\mathbf{p})\right) \quad (3.3)$$

Not all model variables may be time-varying, but regardless of this fact, dim(**m**) can be extremely large for real-world applications (10^6–10^9).

We now return to the data variables. The data variables are obtained by generating one model realization (e.g., using Monte Carlo, see Section 3.10) and applying the forward model representing the physical relationship between the model and the data variable. Here the data consists of hydraulic head data at four locations. The observed data are denoted as $\mathbf{d}_{\text{obs}}$: the head measurements in four wells.

The design of the system will be based on the evolution of the contaminant in the future as the recharge is terminated. Therefore, key prediction variables, generically denoted as **h**, are as follows:

1. The future concentration (over time) of DNAPL in the drinking well
2. The DNAPL arrival time in the critical zone

These can be forward modeled as well (see Figure 3.2) based on models:

$$f_h(\mathbf{m},\mathbf{h}) = 0 \Rightarrow \mathbf{h} = g_h(\mathbf{m}) = g_h\left(\mathbf{m}_{\text{grid}},\mathbf{p}\right) \quad (3.4)$$

3.2. BASIC ALGEBRA

3.2.1. Matrix Algebra Notation

Some basic algebra notations are reviewed in this section, as well as eigenvalues and eigenvectors, which lie at the foundation of multivariate (high dimension) problems in both mathematics and statistics. Ultimately, regardless of the technique or method developed, all operations can be summarized as matrix operations and most of them rely on some form of orthogonalization. Specific to multivariate modeling, an important matrix is the "data matrix," which consists of multiple observations of a random vector of a certain size. "Data" should not be limited to any field data or actual observation or to data variables; such matrix may contain model realizations. For example, one may generate a set of L model realizations with a total amount of model variables N. Throughout the book, unless otherwise stated, we will use N to represent the dimension of "data," whether models, samples, realizations, and observations, while L represents the amount of "data." For example, consider a model realization,

$$\mathbf{m} = (m_1, m_2, \ldots, m_N) \quad (3.5)$$

Notice the notation as follows: bold for vectors and italic for scalar variables. Now also consider that L model realizations have been generated, in our notation:

$$\mathbf{m}^{(\ell)} = \left(m_1^{(\ell)}, m_2^{(\ell)}, \ldots, m_N^{(\ell)}\right),\ \ell = 1, \ldots, L \quad (3.6)$$

Then, a matrix of model realizations becomes

$$\begin{pmatrix} m_1^{(1)} & m_2^{(1)} & \dots & \dots & \dots & m_N^{(1)} \\ \vdots & m_2^{(2)} & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & \vdots \\ m_1^{(L)} & m_2^{(L)} & \dots & \dots & \dots & m_N^{(L)} \end{pmatrix} \tag{3.7}$$

Consider now applying a forward model, for example the data forward model on each model, to generate the data variable outcomes (such as hydraulic heads in our simple case):

$$\mathbf{d}^{(\ell)} = g_d\left(\mathbf{m}^{(\ell)}\right) = g_d\left(m_1^{(\ell)}, m_2^{(\ell)}, \dots, m_N^{(\ell)}\right),\ \ell = 1, \dots, L$$

Then, another matrix can be generated as

$$\begin{pmatrix} d_1^{(1)} & d_2^{(1)} & \dots & \dots & \dots & d_{N_d}^{(1)} \\ \vdots & d_2^{(2)} & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & \vdots \\ d_1^{(L)} & d_2^{(L)} & \dots & \dots & \dots & d_{N_d}^{(L)} \end{pmatrix} \tag{3.8}$$

with N_d the dimension of the data variables, to distinguish it from the dimension of the model variables.

The following notation and conventions are adapted for matrix algebra, including some specific type of matrices:

1. $X = (x_{ij})$ an $N \times L$ matrix consisting of scalars x_{ij}
2. A row vector $\mathbf{x}^T$ with $\mathbf{x}$ a column vector
3. $\mathbf{1}_N = (1, \dots, 1)^T$ a vector of ones with length N
4. $\mathbf{0}_N = (0, \dots, 0)^T$ a vector of zeros with length N
5. a diagonal matrix diag(x_{ii}), $x_{ij}\ \forall\ i \neq j$
6. an identity matrix diag(1, …, 1) = I_N
7. a unity matrix $\mathbf{1}_N\mathbf{1}_N^T$
8. Trace: tr(X)
9. Determinant det(X)

3.2.2. Eigenvalues and Eigenvectors

When multiplying a matrix of size $N \times L$ with a vector of size $L \times 1$ (or simply L), a vector of size $N \times 1$ is obtained. In one specific interpretation, when dealing with Cartesian axis systems (more about spaces and geometries later), such operation can be seen as mapping or projection of a vector in a Cartesian axis system of dimension N into a new Cartesian axis system with dimension L. When $L < N$ then this amounts to a dimension reduction, otherwise a dimension increase.

Consider now the special case where such matrix is a square matrix. For example, a linear relationship between model and data exists (or be assumed):

$$G\mathbf{m} = \mathbf{d} \tag{3.9}$$

Hence, the forward model operator, as a linear model, is expressed in matrix G. We consider for now that the dimension of data and model are the same:

$$N = N_m = N_d \tag{3.10}$$

Now we wish to analyze the properties of G. This is relevant, since the specifics of G will determine how L model realizations are mapped into L data realizations (see Figure 3.3). Since G is a linear operator we do not expect such mapping to change the topology of the space. Indeed, we expect that the model point cloud in Figure 3.3 is stretched/squeezed and rotated, perhaps

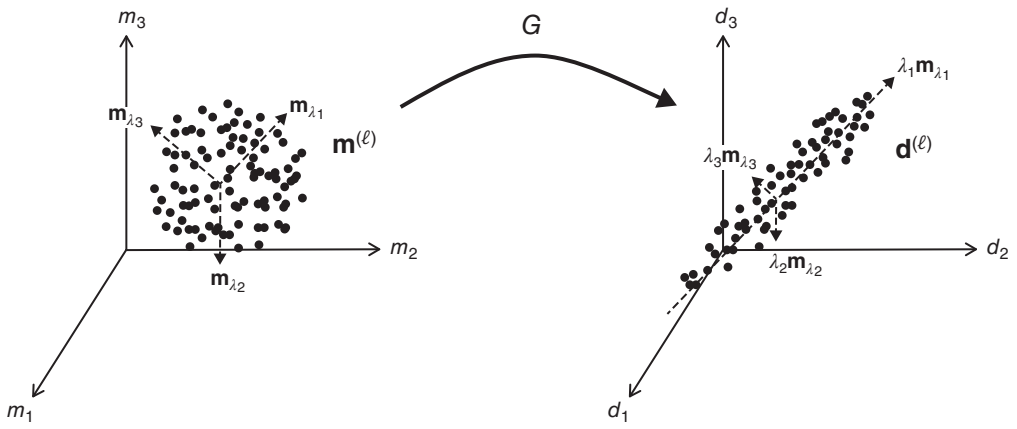


Figure 3.3 A matrix multiplication representing a mapping from one Cartesian axis space to another Cartesian axis space. Eigenvalues and eigenvectors characterize the nature of such transformation.

translated into a new point cloud (the data point cloud). It is, therefore, interesting to study which model realizations (model vectors) are modified/transformed only up to a scalar. These special model realizations are termed eigenvectors of the transformation G.

Formally, consider a square matrix G, if a scalar λ and vector $\mathbf{m}_\lambda$ exists such that

$$G\mathbf{m}_\lambda = \lambda \mathbf{m}_\lambda \tag{3.11}$$

then λ is termed an eigenvalue and $\mathbf{m}_\lambda$ an eigenvector. In this case, eigenvalues can be calculated as roots of and Nth order polynomial $\det(G - \lambda I_N)$; hence, up to N eigenvalues exist $\lambda_1, \ldots, \lambda_N$, for each eigenvalue a corresponding eigenvector exists $\mathbf{m}_{\lambda_1}, \ldots, \mathbf{m}_{\lambda_N}$. The determinant and trace can be written as function of eigenvalues:

$$\det(G) = \prod_n^N \lambda_n \tag{3.12}$$

$$\mathrm{tr}(G) = \sum_n^N \lambda_n \tag{3.13}$$

An intuitive explanation by means of geometric description for trace and determinant is as follows. The determinant of a matrix represents the signed volume change of a unit cube into a parallelepiped after projection with G. The sign indicates how the volume is rotated (clockwise or counterclockwise). The sign and volume change depends on the sign of the eigenvalues as well as their magnitudes, where increases in volume occur when λ are larger than unity in absolute value and decreases for absolute values smaller than unity.

The trace has a geometric interpretation of a change of that volume after projection under an infinitesimal change before projection; hence, the trace determines "how big" the projection is. Both determinant and trace have important application in UQ such as in the analysis of covariance matrices, least-squares methods, or in dimension reduction methods.

3.2.3. Spectral Decomposition

3.2.3.1. Theory. The spectral decomposition or Jordan decomposition links the structure of a matrix to the eigenvalues and the eigenvectors. Each symmetric matrix can be written as

$$A = V \Lambda V^T \tag{3.14}$$

For example, when considering the forward model G of the previous section,

$$G = V \Lambda V^T = \sum_{n=1}^N \lambda_n \mathbf{m}_{\lambda_n} \mathbf{m}_{\lambda_n}^T \tag{3.15}$$

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N) \tag{3.16}$$

$$V = (\mathbf{m}_{\lambda_1}, \ldots, \mathbf{m}_{\lambda_N}) \quad VV^T = I \tag{3.17}$$

If all eigenvalues are positive then

$$G^{-1} = V \Lambda^{-1} V^T \tag{3.18}$$

In the more general case, an operator may not be a square matrix, for example when the data variable dimension is different from the model dimension variable ($N_m \neq N_d$), then each (non-square) matrix with rank r can be decomposed as

$$G = V \Sigma U^T \quad VV^T = UU^T = I_r \tag{3.19}$$

with V a matrix of size $(N_m \times r)$ and U a matrix of size $(N_d \times r)$.

$$\Sigma = \mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r}\right), \quad \lambda_i > 0 \tag{3.20}$$

The values $\lambda_1, \ldots, \lambda_r$ are the nonzero eigenvalues of GG^T and G^TG with V and U containing the corresponding eigenvalues of these matrices.

3.2.3.2. Geometric interpretation. Singular value decomposition (SVD) lies at the heart of many useful methods of UQ covered later in this book. SVD decomposes a matrix (whether a data matrix, a covariance matrix) into simpler, more easily interpretable and meaningful parts. To understand this, again geometrically, we consider that a matrix A is a linear mapping from one Cartesian space to another (see Figure 3.4). Consider the case of a simple matrix

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{3.21}$$

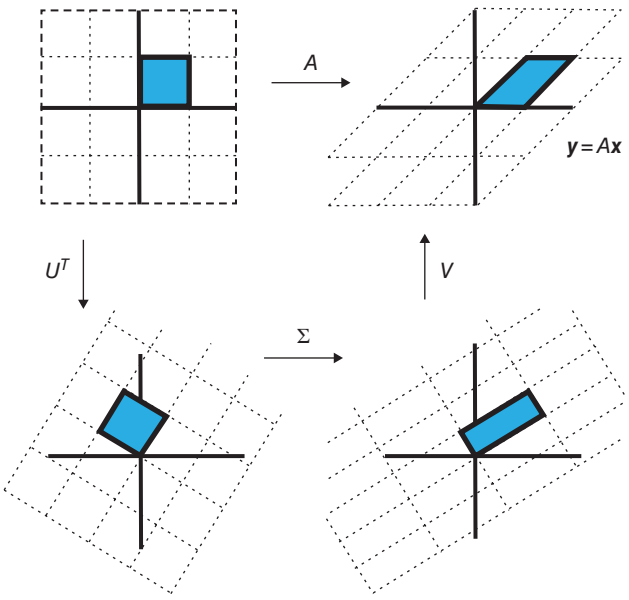


Figure 3.4 Geometric explanation of SVD as a transformation from one orthogonal system to another orthogonal system.

which results in shearing a unit cube when mapping the vectors of that cube using A. SVD allows writing this transformation as a series of affine corrections between one orthogonal space and another orthogonal space. The fact that we map into a space that is also orthogonal means that we can use the usual tools such as Euclidean distance, norms, and so on. To map a vector $\mathbf{x}$ with A is now equivalent to

$$\mathbf{y} = A\mathbf{x} = V\Sigma U^T\mathbf{x} \tag{3.22}$$

which is equivalent to applying a rotation U^T (here by 58.28°), a stretching (Σ), and another rotations V (again by 58.28°). A here is a full rank matrix. If the matrix is not full rank then the mapping (rank = 1) is onto a single vector. However, the formulation is still the same, meaning one can achieve such mapping, always, by means of rotations and stretching (even if stretched to infinity).

3.2.4. Quadratic Forms

Quadratic forms in higher dimensions are a useful way to study properties of data. These forms are often associated with least squares, inverse modeling, or the multi-Gaussian distribution. Calculating the derivative of a quadratic form in the context of optimization (e.g., maximum likelihood method) leads to a linear system of equations that can then be solved with standard techniques. Since the quadratic form is associated with a symmetric matrix, any application that involves such matrix, such as a covariance matrix, relies on these forms.

Consider $\mathbf{x} \in \mathbb{R}^L$, then a quadratic form is built from a symmetric matrix A as follows:

$$Q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} = \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L} a_{\ell\ell'}x_\ell x_{\ell'} \tag{3.23}$$

If $Q(\mathbf{x}) > 0, \quad \mathbf{x} \neq 0$ then the quadratic form is positive definite. A is then positive definite if the corresponding quadratic form is positive definite. Applications occur when A is a covariance matrix of $\mathbf{x}$ (see later) or when A is the matrix of second derivatives on $\mathbf{x}$. In the latter case, the quadratic form measures the curvature in $\mathbf{x}$. The eigenvalues of A determine the shape of the quadratic form. It is easy to show that with λ_ℓ as eigenvalues and V as eigenvectors that

$$\mathbf{x}^T A\mathbf{x} = \sum_{\ell=1}^{L}\lambda_\ell y_\ell^2 \text{ with } \mathbf{y} = V^T\mathbf{x} \tag{3.24}$$

An interesting property of quadratic forms relates to the extrema of these forms. Consider two symmetric matrices A and B, then

$$\max_{\mathbf{x}}\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T B\mathbf{x}} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L = \min_{\mathbf{x}}\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T B\mathbf{x}} \tag{3.25}$$

with λ_1 the eigenvalues of $B^{-1}A$. In the specific case when $\mathbf{x}^T B\mathbf{x}=1, \forall\, \mathbf{x}$ we get

$$\max_{\mathbf{x}}\mathbf{x}^T A\mathbf{x} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L = \min_{\mathbf{x}}\mathbf{x}^T A\mathbf{x} \tag{3.26}$$

This property will be useful later when searching for linear combinations in the data $A\mathbf{x}$ that maximally explain variance (variance = a square form) of that data.

3.2.5. Distances

3.2.5.1. Basics. Distances form an important component of many of the UQ tools presented in this book. Models of the subsurface are usually complex and high dimensional. Hence, representing them mathematically in some extremely high-dimensional Cartesian space is not feasible. In addition, Cartesian spaces may not be the best choice for physical properties (see Chapter 6). We may not at all be interested in the model itself, but we may be interested in the difference between one model and another model. After all, UQ is only meaningful if models "differ" in some sense or another. Of relevance is that they differ in the prediction calculated from them or in some other summary statistics. The mathematical foundation of difference is "distance" and the space created is metric space.

A metric space is a set for which the distances between the members of the set are defined. For example, a set of L model realizations is such a set or the L data variables or response variables calculated from the models is also a set. If we call this set $X = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(L)})$, then the following axioms of distance are formulated for a metric space:

$$\begin{aligned}
&d\left(\mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell')}\right) \geq 0 && \text{non-negativity}\\
&d\left(\mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell')}\right) = 0 \Leftrightarrow \mathbf{x}^{(\ell)} = \mathbf{x}^{(\ell')} && \text{identity}\\
&d\left(\mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell')}\right) = d\left(\mathbf{x}^{(\ell')}, \mathbf{x}^{(\ell)}\right) && \text{symmetry}\\
&d\left(\mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell'')}\right) \leq d\left(\mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell')}\right) + d\left(\mathbf{x}^{(\ell')}, \mathbf{x}^{(\ell'')}\right) && \text{triangular inequality}
\end{aligned} \tag{3.27}$$

Here d is called the distance function. Well-known metric spaces are the real numbers with absolute difference or any Euclidean space with a Euclidean distance defined as

$$d\left(\mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell')}\right) = \sqrt{\left(\mathbf{x}^{(\ell)} - \mathbf{x}^{(\ell')}\right)^T\left(\mathbf{x}^{(\ell)} - \mathbf{x}^{(\ell')}\right)} \tag{3.28}$$

3.2.5.2. Useful Distances for UQ

3.2.5.2.1. Univariate Variables. A wide class of differences are generated based on norms:

$$d\left(\mathbf{x}^{(1)},\mathbf{x}^{(2)}\right)=\left\|\mathbf{x}^{(1)}-\mathbf{x}^{(2)}\right\|_r=\left(\sum_{n=1}^{N}\left|x_n^{(1)}-x_n^{(2)}\right|^r\right)^{1/r} \tag{3.29}$$

Hence, any normed vector space is a metric space. An assumption here is that the components of $\mathbf{x}$ are on the same scale. The Manhattan norm gives rise to the Manhattan distance (also termed $L1$ distance), where the distance between any two vectors is the sum of the differences between corresponding components. The maximum norm gives rise to the Chebyshev distance or chessboard distance.

The norms above are appropriate when dealing with continuous variables; however, they become problematic for categorical variables. Categorical variables may not have ordinality. Consider the following examples of geological sequences:

$$\mathbf{S}_1: D\,F\,F\,E\,D\,E \quad D=\text{delta},\quad F=\text{fluvial},\quad E=\text{estuarine}$$

$$\mathbf{S}_2: F\,D\,F\,E\,F\,E$$

What is a measure of their difference? In the absence of order, there should be no difference between D E and D F. To alleviate this, we need to create indicator variables

$$I_D=\begin{cases}1 & \text{if } S=D\\ 0 & \text{else}\end{cases}\quad I_E=\begin{cases}1 & \text{if } S=E\\ 0 & \text{else}\end{cases}\quad I_F=\begin{cases}1 & \text{if } S=F\\ 0 & \text{else}\end{cases} \tag{3.30}$$

and hence an appropriate distance is

$$d(\mathbf{S}_1,\mathbf{S}_2)=\frac{1}{3}\sum\nolimits_{s\in\{D,E,F\}}I_s \tag{3.31}$$

3.2.5.2.2. Hausdorff Distance. The Hausdorff distance is popular in image analysis for measuring the dissimilarity between two sets of data $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$ [*Huttenlocher et al.*, 1993]. Individual elements of $\mathbf{S}^{(1)}$ are denoted as $s_i^{(1)}$ and individual elements of $\mathbf{S}^{(2)}$ are denoted as $s_i^{(2)}$. $\mathbf{S}^{(1)}$and $\mathbf{S}^{(2)}$ are deemed similar if each point in one set is close to all other points in the other set. Consider two objects in Figure 3.5. What is the distance between these objects? First, each has been rasterized into a set of points. Consider first calculating $d\left(s_i^{(1)},\mathbf{S}^{(2)}\right)$ between one point and the set. This distance is defined as the minimum:

$$d\left(s_i^{(1)},\mathbf{S}^{(2)}\right)=\min_j d\left(s_i^{(1)},s_j^{(2)}\right) \tag{3.32}$$

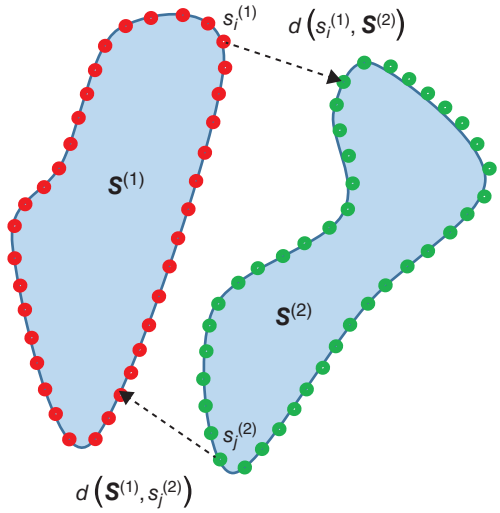


Figure 3.5 Creating a Hausdorff distance between two objects that are discretized with points.

Then, to calculate $d(\mathbf{S}^{(1)},\mathbf{S}^{(2)})$, *Dubuisson and Jain* [1994] propose to average over the points in set $\mathbf{S}^{(1)}$:

$$d\left(\mathbf{S}^{(1)},\mathbf{S}^{(2)}\right)=\frac{1}{N_1}\sum_{i=1}^{N_1}\min_j d\left(s_i^{(1)},s_j^{(2)}\right) \tag{3.33}$$

The resulting measure is not symmetric $d(\mathbf{S}^{(1)},\mathbf{S}^{(2)})\neq d(\mathbf{S}^{(2)},\mathbf{S}^{(1)})$, hence not a distance, so technically we cannot use the notation d. *Dubuisson and Jain* [1994] propose to use the maximum to symmetrize the distance of Eq. (3.34)

$$\begin{aligned}d\left(\mathbf{S}^{(1)},\mathbf{S}^{(2)}\right)&=d\left(\mathbf{S}^{(2)},\mathbf{S}^{(1)}\right)\\&=\max\left[\frac{1}{N_1}\sum_{i=1}^{N_1}\min_j d\left(s_i^{(1)},s_j^{(2)}\right),\frac{1}{N_2}\sum_{j=1}^{N_2}\min_i d\left(s_i^{(1)},s_j^{(2)}\right)\right]\end{aligned} \tag{3.34}$$

Figure 3.6 shows an example of how the Hausdorff distance outperforms the Euclidean distance in discriminating between objects.

3.2.5.2.3. Distances Based on Transformations. Simple Euclidean distances between images, or spatial models, often are not very informative about their actual difference. Consider a simple case of three simple images with one line (e.g., a fracture or fault). In the second image, the line is slightly offset, hence there are no overlapping pixels, while the third image is orthogonal to the first image, and therefore it has at least one location in common. The Euclidean distance cannot capture the significant similarity between images 1 and 2.

A solution is to transform the categorical variable into a continuous variable that informs the distance to the edge

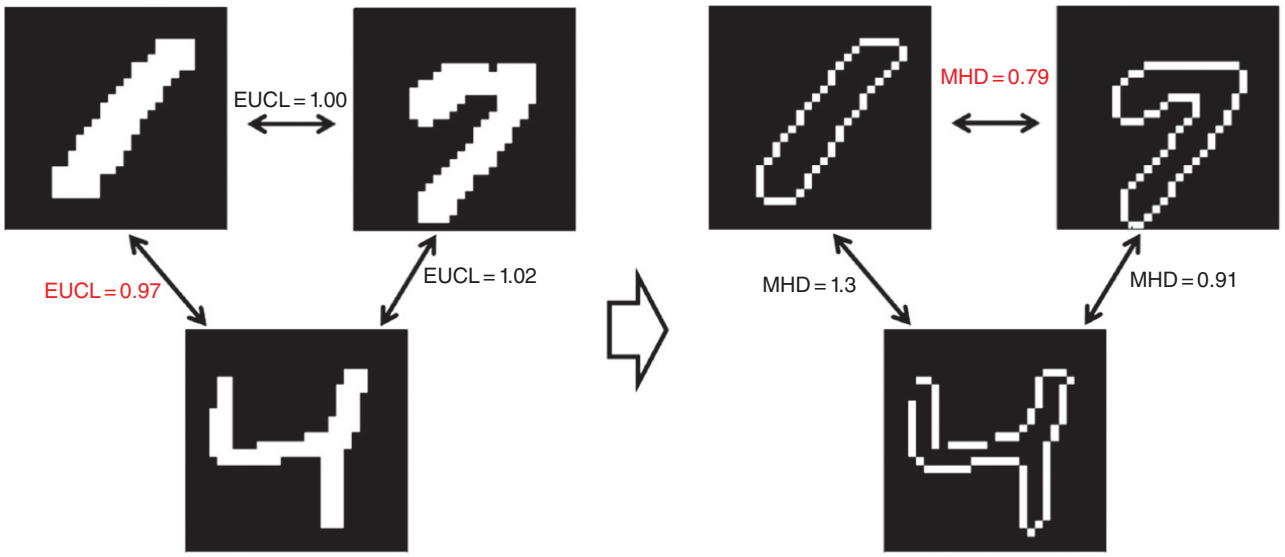


Figure 3.6 Evaluation of the modified Hausdorff distance for images of numbers 1, 4, and 7. Modified Hassdorf distance (MHD) requires the rasterization of the images into points sets (shown on the right). The smallest MHD is observed between images of 1 and 7, which is consistent with visual inspection. This is not the case for the pixel by pixel Euclidean distance (left).

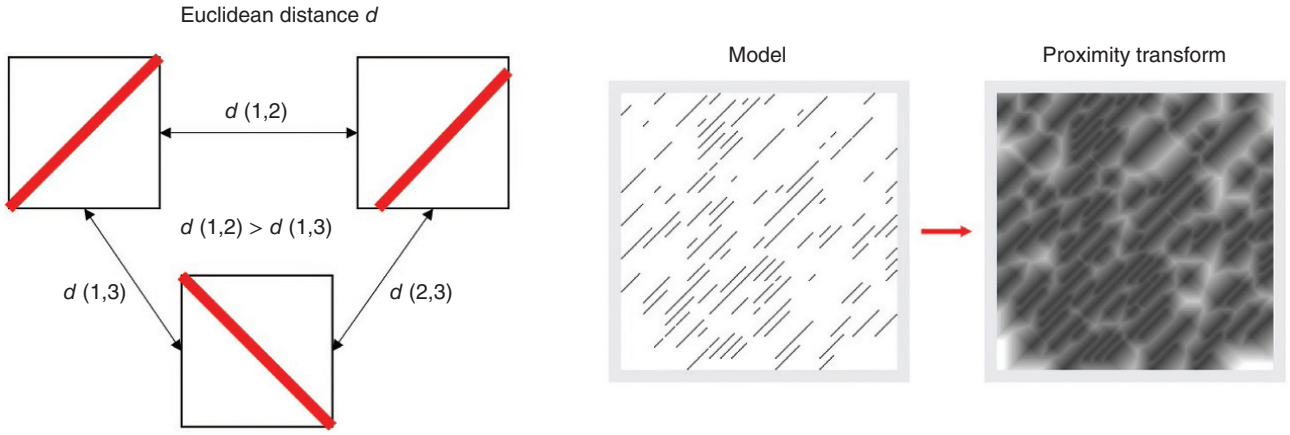


Figure 3.7 The proximity transforms to measure differences in images constituted by discrete objects.

of the feature of interest. This proximity transform results in a "distance map" as shown Figure 3.7. It results in a transformed variable where the grayscale levels indicate the distance to the diagonal black object. The computation of a distance between both distance maps, using for example a Euclidean norm, is then more meaningful than the distance between categorical patterns.

3.2.5.2.4. Distances Between Distributions. Is the prior distribution different from the posterior? Is the posterior distribution generated by a Markov chain Monte Carlo method similar to the theoretical posterior, or the posterior of another method? Is the histogram of one model realization (e.g., porosity) significantly different from another realization? All these questions call for a measure of difference between distributions. The statistical literature offers various methods to test whether two distributions are statistically significantly different. In this section, we will limit ourselves to comparing univariate distributions. In the application chapters, we will see how comparisons between multivariate distributions are achieved by comparing distributions of orthogonal components of the random vector in question. Consider to that end two discrete probability distributions represented by the discrete probabilities p_k, q_k, $k = 1, \ldots, K$. A well-known distance is the chi-squared distance

$$d_{\chi^2}(\mathbf{p},\mathbf{q}) = \frac{1}{2}\sum_{k=1}^{K}\frac{(p_k - q_k)^2}{(p_k + q_k)} \tag{3.35}$$

This distance may underweight small differences because of the square, but it is symmetric. Another measure of difference related to information theory is the Kullback–Liebler (KL) divergence

$$\mathrm{dif}_{KL}(\mathbf{p},\mathbf{q}) = \sum_{k=1}^{K} p_k \log\frac{p_k}{q_k} \tag{3.36}$$

which is the expected value of the logarithmic differences. This measure is not symmetric; it emanates from information theory where information is optimally coded by assigning the smallest code to the most frequent letter/message. This measure can be interpreted as the expected extra message-length that is communicated if a code optimal for some assumed distribution (**q**) is used, compared to a code that is based on the underlying, unknown true distribution (**p**). The symmetric form of the *KL* difference is the Jensen–Shannon divergence:

$$d_{JS}(\mathbf{p},\mathbf{q}) = \frac{1}{2}\mathrm{dif}_{KL}(\mathbf{p},\mathbf{q}) + \frac{1}{2}\mathrm{dif}_{KL}(\mathbf{q},\mathbf{p}) \tag{3.37}$$

Note that the continuous form of the *KL* distance is

$$\mathrm{dif}_{KL}(p(x),\, q(x)) = \int p(x)\log\frac{p(x)}{q(x)}dx \tag{3.38}$$

with $p(x)$ and $q(x)$ densities. Another distance is the earth movers distance (EMD) where pdfs are seen as two piles of material. The EMD is then defined as the minimum cost of turning one pile into the other. The cost is defined as the amount of material moved times the distance by which it is moved. A last example based on pdfs is the Bhattacharyya distance:

$$d_{BC}(\mathbf{p},\mathbf{q}) = -\log\left(\sum_{k=1}^{K}\sqrt{p_k q_k}\right) \tag{3.39}$$

In terms of cdfs one can use the $L1$ norm, which is basically the area between the two cdfs.

3.3. BASICS OF UNIVARIATE AND MULTIVARIATE PROBABILITY THEORY AND STATISTICS

In this section, we present some basic elements of multivariate probability theory, mostly to cover notation and conventions and for those who need a brief refresher.

3.3.1. Univariate Transformations

The Box–Cox transform is a procedure for transforming data into a normal shape. It uses a single parameter λ, such that for each sample $x^{(\ell)}$

$$x^{(\ell)}_{\mathrm{trans}} = \begin{cases} \dfrac{\left(x^{(\ell)}\right)^{\lambda} - 1}{\lambda}, & \lambda \neq 0 \\ \ln x^{(\ell)}, & \lambda = 0 \end{cases} \tag{3.40}$$

λ can range from -5 to 5, with a special case when $\lambda = 0$, which is known as the log-transform. To determine which value of λ to be used, we can search over the range for the value that maximizes the correlation between $x^{(\ell)}_{\mathrm{trans}}$ and a theoretical normal distribution. The Box–Cox transformation does not guarantee that the resulting distribution is actually normal, so it is essential to perform a check after the transformation. The Box–Cox transform can be applied only to positive data, so it may be necessary to add a constant to $x^{(\ell)}$ to ensure this.

Another useful transform that is useful when $x^{(\ell)}$ varies from 0 to 1 such as for proportions or percentages.

$$x^{(\ell)}_{\mathrm{trans}} = \sin^{-1}\left(\sqrt{x^{(\ell)}}\right) \tag{3.41}$$

The result of the transform is given in radians ranging from $-\pi/2$ to $\pi/2$. The arcsin transform is helpful when the variance of the variable is uneven (smaller near 0 and 1) by spreading the variance over the entire range and can make the variable more normal.

The rank transform is a common way to transform into any distribution type, such as for example a normal score transform. It also allows any easy back-transformation. We first rank the data

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(L)} \tag{3.42}$$

the subscript indicating the rank. The cumulative frequency for each ranked sample:

$$p_\ell = \frac{\ell}{L} - \frac{1}{2L} \tag{3.43}$$

then for any cumulative distribution function G (such as a normal distribution)

$$x^{(\ell)}_{\mathrm{trans}} = G^{-1}(p_\ell) \tag{3.44}$$

3.3.2. Kernel Density Estimation

The goal of density estimation is to estimate a probability density function using only samples drawn from it. The simplest form of density estimation is the histogram. By dividing sample spaces into a fixed number of equally sized bins and counting the fraction of samples that fall within each bin, an estimate of the density over the bin interval is obtained. This histogram is straightforward to compute, but it results in discontinuities in the estimated density because of the discrete nature of the bins. Furthermore, as the dimension of the space increases, the number of bins increases exponentially. Another

way offers itself by density estimation based on the kernel method and is widely used in the later chapters.

Based on Parzen windows, which places a bin centered at each sample, the kernel density estimate at x is then the sum of the number of bins that encompass it. Therefore, for a given value x, the density is expressed as

$$\hat{f}(x) = \frac{1}{L}\sum_{\ell=1}^{L} K\left(\frac{x^{(\ell)} - x}{w}\right) = \frac{1}{L}\sum_{\ell=1}^{L} K\left(\frac{z_\ell}{w}\right) \tag{3.45}$$

where z_ℓ is the distance between the location at which the density is being evaluated and sample point $x^{(\ell)}$, and K is the kernel or the shape of the bin. For Parzen windows, the kernel is a rectangular function, and the width is set by the w parameter. This approach fixes the number of bins to the number of samples, which alleviates the problem of exponentially increasing number of bins in high dimensions. However, Parzen windows do not address the issue of discontinuities because of the shape of the kernel. A smoothly varying function such as the Gaussian kernel is more frequently used:

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) \tag{3.46}$$

Other popular kernels are the uniform, triangular, and Epanechnikov kernels. The method is named *kernel density estimation*. w is the smoothing parameter or *bandwidth* of the estimator. In higher dimensions, the bandwidth is the covariance matrix of the Gaussian kernel. The bandwidth controls the extent of the influence each sample has on estimating the density. The choice of bandwidth has a very strong influence on the resulting density estimate. A small bandwidth will result in "spiky" density estimates, while a large choice will over smooth the estimate and obscure its structure.

A popular choice for w is Silverman's rule of thumb, which assumes that the underlying distribution is Gaussian. The diagonal components of the matrix are

$$w_n = \left(\frac{4}{N+2}\right)^{\frac{1}{N+2}} L^{-\frac{1}{N+4}} \hat{\sigma}_n \tag{3.47}$$

where $\hat{\sigma}_n$ is an estimate of the standard deviation of the n-th variate, and N is the dimension of the problem.

In theory, kernel density estimation can be extended to any number of dimensions. The kernel function in Eq. (3.47) is simply extended to higher-dimensional functions. However, in practice because of the curse of dimensionality, the number of samples required for accurate estimation grows exponentially with dimension. *Silverman* [1986] provides an optimistic upper bound on the number of dimensions as five, before the number of required samples for accurate joint density estimation often becomes impractically large. Nonetheless, smoothing techniques provide a powerful way of gaining insight into complex distributions. This additional flexibility does, however, come with the challenging task of specifying the bandwidth, as well as the limitation of only working effectively in low dimensions.

3.3.3. Properties of Multivariate Distributions

In multivariate statistics, we study random vectors, generically written as $\mathbf{X} = (X_1, \ldots, X_N)$. The stochastic variation of these random vectors is fully described by the joint cumulative distribution

$$P(X_1 \le x_1, X_2 \le x_2, \ldots, X_N \le x_N) = F(x_1, x_2, \ldots, x_N) \tag{3.48}$$

or the corresponding joint (multivariate) density function

$$f(x_1, x_2, \ldots, x_N) = \frac{\partial^N F(x_1, x_2, \ldots, x_N)}{\partial x_1 \partial x_2 \ldots \partial x_N} \tag{3.49}$$

From the full multivariate distribution, one can deduce any marginal distribution, such as a univariate or bivariate distribution

$$\begin{aligned} F(x_n) &= \int dx_1 \cdots \int dx_{n-1} \int dx_{n+1} \cdots \int f(x_1, \ldots, x_N) dx_N \\ F(x_1, x_2) &= \int dx_3 \int dx_4 \cdots \int f(x_1, \ldots, x_N) dx_N \end{aligned} \tag{3.50}$$

or any conditional distribution

$$\begin{aligned} F(x_n | X_{n'} = x_{n'} \forall n' \ne n) &= P(X_n \le x_n | \, X_{n'} = x_{n'} \forall n' \ne n) \\ &= \frac{\int_{-\infty}^{x_{n'}} f(x_1, \ldots, x_{n'-1}, x'_{n'}, x_{n'+1}, \ldots, x_N) dx'_{n'}}{f(x_1, \ldots, x_{n'-1}, x_{n'+1}, \ldots, x_N)} \end{aligned} \tag{3.51}$$

The problem in reality is that very few analytical expressions exist for the joint multivariate distribution (except for the Gaussian, see later). The focus instead lies on lower-order statistics, such as the bivariate distributions from which then moments such as the variogram γ and covariance, cov, can be derived (and estimated with data)

$$\begin{aligned} \gamma(X_n, X_{n'}) &= \frac{1}{2} E(X_n - X_{n'})^2 \\ &= \frac{1}{2} \int\int (x_n - x_{n'})^2 f(x_n, x_{n'}) dx_n dx_{n'} \end{aligned} \tag{3.52}$$

$$\begin{aligned} \mathrm{cov}(X_n, X_{n'}) &= E[(X_n - E[X_n])(X_{n'} - E[X_{n'}])]^2 \\ &= \int\int (x_n - E[X_n])(x_{n'} - E[X_{n'}]) f(x_n, x_{n'}) dx_n dx_{n'} \end{aligned} \tag{3.53}$$

From the latter we define correlation as

$$\rho(X_n, X_{n'}) = \frac{\mathrm{cov}(X_n, X_{n'})}{\sqrt{\mathrm{var}(X_n)\,\mathrm{var}(X_{n'})}} \tag{3.54}$$

When the $(X_1, \ldots, X_N)$ are (globally) independent then

$$f(x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} f(x_n) \tag{3.55}$$

Global independency entails also pair-wise independence but not vice versa. Pair-wise independence means that any bivariate distribution becomes a product of marginal, moreover then:

$$\rho(X_n, X_{n'}) = 0 \ \forall n, n' \tag{3.56}$$

Equation (3.56) entails linear independence only. When the relationship is nonlinear, then Eq. (3.56) may still hold, even if pair-wise dependency exists. Another form of independence is conditional independence, which in the univariate case becomes

$$f(x_n, x_{n'} | x_{n''}) = f(x_n | x_{n''}) f(x_{n'} | x_{n''}) \tag{3.57}$$

or equivalently

$$f(x_n | x_{n'}, x_{n''}) = f(x_n | x_{n'}) = f(x_n | x_{n''}) \tag{3.58}$$

This concept can be extended to any mutually exclusive subset of random variables that comprise the random vector $\mathbf{X}$. The concept of conditional independence is used throughout UQ. For example, one may have various types of data that inform the subsurface $\mathbf{d}_1$, $\mathbf{d}_2$, …, $\mathbf{d}_n$. These data may be from different origin, such as one data source from geophysics, another from drilling, or yet another from testing the subsurface formation. The conditional independence assumption that is used is then as follows:

$$f(\mathbf{d}_1, \mathbf{d}_2, \ldots | \mathbf{m}) = \prod\nolimits_i f(\mathbf{d}_i | \mathbf{m}) \tag{3.59}$$

Basically this means that if we knew the real earth, then these data sets can be treated as independent. In other words, knowing the real earth is enough to model each data set separately through individual likelihoods $f(\mathbf{d}_i|\mathbf{m})$. This appears quite reasonable for data sets that are very different, for example whose forward models capture very different physics.

3.3.4. Characteristic Property

Recall that among all outcomes of a random variable, the expected value is known to minimize the average square error (essentially a variance):

$$E[X] = \min_{x_0} E\left[(X - x_0)^2\right] \tag{3.60}$$

This property lies at the heart of all least-square method: the expectation is the best least-square estimate of an unknown RV. Similarly, consider the conditional mean as a $N-1$ dimensional function

$$E[X_n | X_{n'} = x_{n'}, \forall n' \neq n] = \psi(x_{n'}, n' \neq n) \tag{3.61}$$

Among all $N-1$ variate functions, the conditional expectation in Eq. (3.61) minimizes the estimation variance

$$\psi(x_{n'}\, n' \neq n) = \min_{\psi_0(x_{n'},\ n' \neq n)} E\left[(X_n - \psi_0(x_{n'}, n' \neq n) | X_{n'} = x_{n'} \forall n' \neq n)^2\right] \tag{3.62}$$

or any function that minimizes a variance (least square) is an expectation, whether conditional or unconditional. This property will be used throughout the book when dealing with least squares, Gaussian processes (see Section 3.7), linear inverse problems, and so on.

3.3.5. The Multivariate Normal Distribution

The multivariate normal is a very popular model in multivariate statistics as well as in UQ. The reason lies in the mathematical convenience of this model, its arranged marriage will least-square and maximum likelihood estimation methods as well as linear modeling. When $\mathbf{X} = (X_1, \ldots, X_N)$ is multivariate normal then

$$f(x_1, \ldots, x_N) = \frac{1}{\sqrt{(2\pi)^N \det(C)}} \exp\left(-\frac{1}{2}(\mathbf{x} - E[\mathbf{X}])^T C^{-1} (\mathbf{x} - E[\mathbf{X}])\right) \tag{3.63}$$

One notices the quadratic form in the exponent (basically a second-order polynomial in $\mathbf{x}$). This form is relevant because the log of the density is quadratic, and hence the derivative is linear. C is the covariance matrix with elements

$$[C]_{nn'} = c_{nn'} = \mathrm{cov}(X_n, X_{n'}), \quad n, n' = 1, \ldots, N \tag{3.64}$$

$(\mathbf{x} - E[\mathbf{X}])^T C^{-1} (\mathbf{x} - E[\mathbf{X}])$ is also termed the Mahalanobis distance: a distance of a point $\mathbf{x}$ from some center $E[\mathbf{X}]$. It can be generalized to a distance as

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T C^{-1} (\mathbf{x} - \mathbf{x}') \tag{3.65}$$

This distance accounts for the correlation that may exist. If no correlation exists then $C = I$, and we get the Euclidean distance. The more $\mathbf{x}$ is correlated with $\mathbf{x}'$, the less that component will contribute to the distance (they look closer, because they are more similar). The Mahalanobis transformation renders the elements (linearly) independent:

$$\mathbf{Y} = C^{-1/2} (\mathbf{X} - E[\mathbf{X}]) \tag{3.66}$$

The elements of $\mathbf{Y}$ have no (linear) correlation.

Some useful properties of the multivariate normal are as follows:

1. All $N-n$ variate distribution are multivariate normal.
2. All conditional distribution are multivariate normal.

3. In case of a univariate conditional distribution, we find that the corresponding conditional expectation are linear functions of the conditioning values, for example

$$E[X_n|X_{n'} = x_{n'}, \forall n' \neq n] = \sum_{n' \neq n} w_{n'} x_{n'} \tag{3.67}$$

or for any subset of conditioning values.

4. The joint distribution of any subset of **X** and linear combinations of another mutually exclusive subset is also multivariate normal, for example

$$\left(\sum_{n' \neq n} w_{n'} X_{n'}, X_n \right) = \left(\hat{X}_n, X_n \right) \sim \text{bivariate normal} \tag{3.68}$$

This results in the so-called conditional unbiasedness property

$$E\left[X_n | \hat{X}_n = x \right] = x \tag{3.69}$$

These properties make the multivariate Gaussian useful in many applications. However, assuming multivariate Gaussian models is not without risk. First, the assumption can rarely be verified with actual data, simply because of the lack of data in higher dimensions. More importantly, assuming a multivariate Gaussian distribution imposes a higher-order dependency structure that goes beyond the covariance defining that distribution. In spatial modeling (where the X_n are properties in a grid), limitations of this distribution are well known [*Gómez-Hernández and Wen*, 1998; *Zinn and Harvey*, 2003; *Feyen and Caers*, 2006]. More specifically, the maximum entropy property [*Journel and Deutsch*, 1993] entails that extremes become rapidly uncorrelated. Additionally, the multivariate distribution is clearly tied to linear modeling; hence, any relationships between the X_n that is not linear becomes problematic.

3.4. DECOMPOSITION OF DATA

3.4.1. Data Spaces

Subsurface models are complex, and data can be extensive, such as for example seismic data; hence, forms of dimension reduction are critical to render such complex problems manageable. Dimension reduction lies at the foundation of many of the methods covered in subsequent chapters. Here we provide the very foundation of most dimension reduction techniques.

Consider again the "data matrix" of size $N \times L$, generically written as

$$X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & \cdots & \cdots & x_N^{(1)} \\ \vdots & x_2^{(2)} & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & \vdots \\ x_1^{(L)} & x_2^{(L)} & \cdots & \cdots & \cdots & x_N^{(L)} \end{pmatrix} \tag{3.70}$$

This matrix can be viewed in two ways: row by row (per "data sample") or column by column (per each dimension of the "data sample"). For example, if the data matrix stores model realizations, then we can look either at the collection of model (rows) or the samples of each individual variable of the model (columns). Hence, from a geometric point of view, one can take two views and create two alternative Cartesian axis systems (see Figure 3.8).

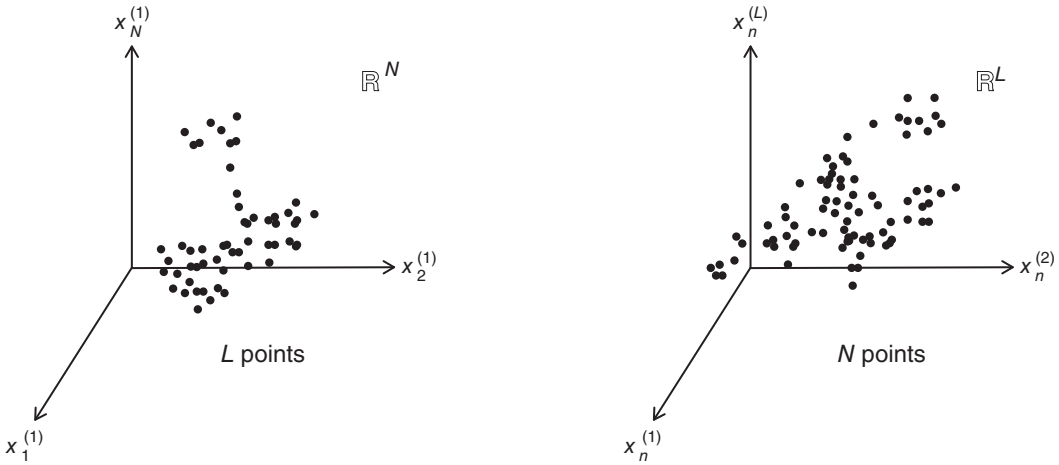


Figure 3.8 Two views of the same data matrix.

1. (Sample dimension) Space $\mathbb{R}^N$: The space containing vectors $\mathbf{x}_N^{(\ell)} = \left(x_1^{(\ell)}, x_2^{(\ell)}, \ldots, x_N^{(\ell)}\right)$, a cloud of $\ell = 1, \ldots, L$ points.

2. (Sample size) Space $\mathbb{R}^L$: The space containing vectors $\mathbf{x}_n^{(L)} = \left(x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(L)}\right)$, a cloud of $n = 1, \ldots, N$ points.

While both spaces convey the same information, choosing which axis system to work with (model, sample, fit, regress) is of critical importance to many of the methods in this book. In fact, we will illustrate several dualities between these two spaces and the techniques that are applied in such spaces. The choice of the axis system largely depends on the dimension of the space. Clearly, a low-dimensional space with a lot of points in the point cloud is easier to manage than a high-dimensional space with very few. High dimensions are challenging because they become very rapidly "empty" as dimension increases. Consider a simple example of a circle within a square and calculate the ratio between the areas occupied by the circle over the area occupied by the square; consider increasing the dimension and calculating the same ratio, now between hypercube and hypersphere. In Figure 3.9 one notices that the ratio is basically zero after dimension 10. In other words, an exponential increase in volume exists with each dimension added to a Cartesian space. This also means that an exponentially increasing amount of samples is needed to cover high-dimensional spaces as one would cover/sample low-dimensional spaces. One characteristic of uncertainty quantification is that subsurface models are complex, spatial, or spatiotemporal; hence it is of very high dimension N. As shown in Chapter 1, the key target variables, on which decisions are made and whose uncertainty is most relevant, are often of much lower dimension than the models. For quantifying their uncertainty (as simple as a volume for example), much less samples L are needed. For most UQ problems, certainly, those treated in Chapter 1, we can safely state that

$$L \ll N \tag{3.71}$$

This is not an observation without considerable consequence. This property is different from data problems that involve people (e.g., Facebook, Google) where the sample size is very large (millions/billions) and the model dimensions (e.g., people's preference) are much smaller. Hence, blindly applying data science in these areas to the problems in this book will be ineffective and inefficient.

3.4.2. Cartesian Space of Size *L*: The Sample Size

In $\mathbb{R}^L$ we need to embed N data points, when considering the data matrix of Eq. (3.70). To reduce dimension, we need to project points into a lower-dimensional space. A straightforward way would be just to ignore a model realization, but this would lead to too much of a loss of information, and potential removal of important realizations from the analysis. Instead, we attempt to project into a lower-dimensional space by minimizing the loss of information caused by such projection. Consider, therefore, first projecting the cloud into one dimension, in other words, a line. Since a line can be defined through a unit vector, $\mathbf{u}_1$, $\|\mathbf{u}_1\| = 1$, we need to find the "optimal" $\mathbf{u}_1$ that best represents the L-dimensional cloud. This is illustrated in Figure 3.10. The coordinate of a projection of a point is

$$\left(\mathbf{x}_N^{(\ell)}\right)^* = \left(\mathbf{x}_N^{(\ell)}\right)^T \mathbf{u}_1 \tag{3.72}$$

As measure of "loss of information" due to such projection, we use a least-square formulation and minimization:

$$\sum_{\ell=1}^{L} \left\| \left(\mathbf{x}_N^{(\ell)}\right)^* - \mathbf{x}_N^{(\ell)} \right\|^2 \tag{3.73}$$

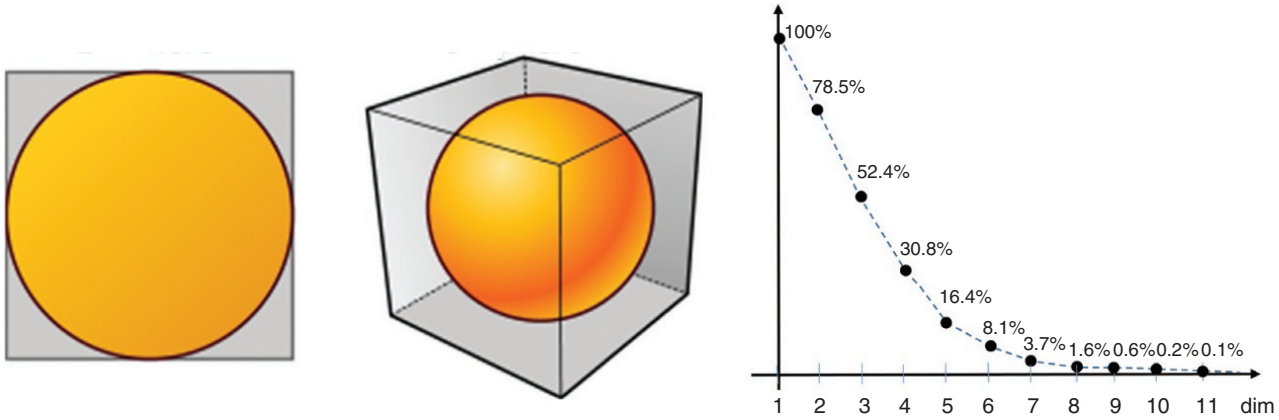


Figure 3.9 Curse of dimensionality, the ratio of hypersphere with hypercube. Space becomes virtually empty after dimension 10. Modified from Wikipedia.

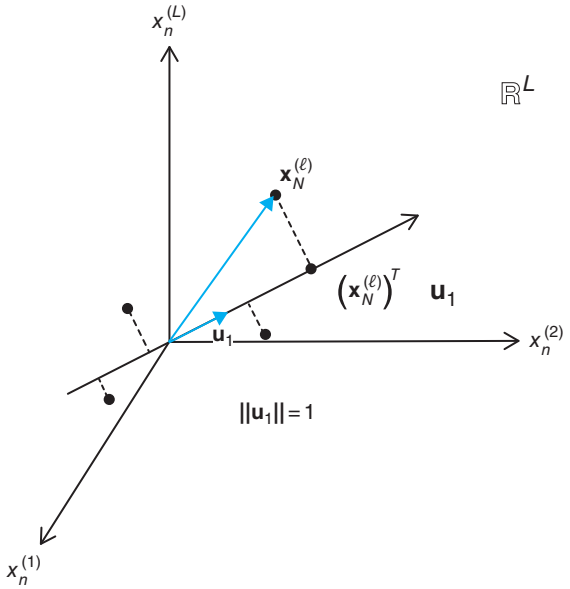


Figure 3.10 Projection of data onto an eigenvector.

The problem of minimizing Eq. (3.73) is equivalent to maximizing

$$\sum_{\ell=1}^{L} \left\| \left(\mathbf{x}_N^{(\ell)}\right)^* \right\|^2 \tag{3.74}$$

which is a convenience resulting from using a square loss function. Because

$$\begin{pmatrix} \left(\mathbf{x}_N^{(1)}\right)^* \\ \left(\mathbf{x}_N^{(2)}\right)^* \\ \vdots \\ \left(\mathbf{x}_N^{(L)}\right)^* \end{pmatrix} = \begin{pmatrix} \left(\mathbf{x}_N^{(1)}\right)^T \mathbf{u}_1 \\ \left(\mathbf{x}_N^{(2)}\right)^T \mathbf{u}_1 \\ \vdots \\ \left(\mathbf{x}_N^{(L)}\right)^T \mathbf{u}_1 \end{pmatrix} = X\mathbf{u}_1 \tag{3.75}$$

maximizing Eq. (3.74) is equivalent to solving

$$\max_{\mathbf{u}_1} (X\mathbf{u}_1)^T (X\mathbf{u}_1) = \max_{\mathbf{u}_1} \mathbf{u}_1^T X^T X \mathbf{u}_1 \text{ subject to } \|\mathbf{u}_1\| = 1 \tag{3.76}$$

According to our previous analysis about quadratic forms such as this, we know that the vector that maximizes Eq. (3.76) is the first eigenvector of X^TX associated with the first eigenvalue.

An important observation is made when the data is centered, meaning that $\sum_{\ell=1}^{L} \frac{\mathbf{x}_N^{(\ell)}}{L} = 0$. Then $1/L\ X^TX$ is the empirical covariance between the L realizations consisting of N variables. Hence, in making such projection, we are seeking to make use of correlations among the variables in X. Obviously, if that covariance is zero (and variance some constant), then we cannot reduce the problem to a lower-dimensional problem, all realizations (L) will need to be used.

The above analysis extends to projections in more than one dimension. The solution is simply found by calculating the eigenvalues λ_n, $n = 1, \ldots, N$ and corresponding eigenvectors $\mathbf{u}_n$ of the matrix X^TX, with its eigenvalues ranked from large to small. Note that in our type of problem X^TX of size $N \times N$ is very large. Directly calculating eigenvalues on X^TX may be impossible for large problems. We will see later that ways around this problem exist.

3.4.3. Cartesian Space of Size N: Sample Dimension

Now we turn to the space $\mathbb{R}^N$. We need to embed L data realizations. Hence, any reduction in this space means reducing the number of variables. Clearly, just ignoring individual variables will usually not be efficient, instead we will, as before, rely on projections; basically, linear combinations of the variables in question. The solution to this projection problem is exactly the same as in the previous section, but now replace X by X^T, namely

$$\begin{aligned} \sum_{n=1}^{N} \left\| \left(\mathbf{x}_n^{(L)}\right)^* \right\|^2 &= \max_{\mathbf{v}_1} (X\mathbf{v}_1)(X\mathbf{v}_1)^T \\ &= \max_{\mathbf{v}_1} \mathbf{v}_1^T XX^T \mathbf{v}_1 \text{ subject to } \|\mathbf{v}_1\| = 1 \end{aligned} \tag{3.77}$$

Hence, we seek the eigenvalues μ_ℓ, $\ell = 1, \ldots, L$ and corresponding eigenvectors v_ℓ of XX^T from largest to smallest to create projections in lower dimensions. Note that the rank

$$r = \text{rank}\left(XX^T\right) = \text{rank}\left(X^TX\right) = \text{rank}(X) \tag{3.78}$$

In other words, we cannot obtain a dimension larger than r. While X^TX is related to the covariance of data variables, calculated from data realizations, XX^T is related to the dot-product of data realizations calculated from data variables. Recall that the dot-product is the projection (a length is a scalar) of one vector onto another vector or, in this case, a projection of one (data) sample onto another (data) sample. Since we have L samples one can calculate $L \times L$ of such dot-products; hence, the dot-product matrix is, in our context, much smaller in size than the covariance matrix.

3.4.4. Relationship Between Two Spaces

Because both spaces represent the exact same information, but each with different projections into lower dimensions, a relationship must exist between these two projections. Consider the eigenvectors and eigenvalues in $\mathbb{R}^N$

$$XX^T\mathbf{v}_\ell = \mu_\ell \mathbf{v}_\ell \quad \ell \le r \tag{3.79}$$

By multiplying each side with X^T, we find

$$\left(X^T X\right)\left(X^T \mathbf{v}_\ell\right) = \mu_\ell \left(X^T \mathbf{v}_\ell\right) \quad \ell \le r \tag{3.80}$$

Hence, each eigenvector $\mathbf{v}_\ell$ of $(X^T X)$ in $\mathbb{R}^N$ corresponds to the eigenvector $X^T \mathbf{v}_\ell$ in $\mathbb{R}^L$, or

$$\mathbf{u}_\ell \sim X^T \mathbf{v}_\ell \tag{3.81}$$

and associated with the same eigenvalue μ_ℓ. Every non-zero eigenvalue of XX^T is also an eigenvalue of $X^T X$. Similarly, when multiplying

$$X^T X \mathbf{u}_n = \lambda_n \mathbf{u}_n \quad n \le r \tag{3.82}$$

Therefore, each eigenvector $\mathbf{u}_n$ of XX^T in $\mathbb{R}^L$ corresponds to the eigenvector $X\mathbf{u}_n$ in $\mathbb{R}^N$, or

$$\mathbf{v}_n \sim X \mathbf{u}_n \tag{3.83}$$

Since the eigenvectors $\mathbf{u}$ and $\mathbf{v}$ need to be unit vectors, the proportionality constant in both cases is $1/\sqrt{\lambda_k} = 1/\sqrt{\mu_k}$.

This leads to stating the following duality: for a data matrix X of size $N \times L$ with rank r, the eigenvalues (number of eigenvalues $\le r$) of $X^T X$ and XX^T are the same and the eigenvectors are related as follows:

$$\begin{aligned} \mathbf{u}_k &= \frac{1}{\sqrt{\lambda_k}} X^T \mathbf{v}_k \quad k \le r \\ \mathbf{v}_k &= \frac{1}{\sqrt{\lambda_k}} X \mathbf{u}_k \end{aligned} \tag{3.84}$$

Additionally, it can be shown that with $U = (\mathbf{u}_1 \ \mathbf{u}_2 \cdots \mathbf{u}_r)$, $V = (\mathbf{v}_1 \ \mathbf{v}_2 \cdots \mathbf{v}_r)$, and $\Sigma = \mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r}\right)$, we obtain the SVD of X:

$$X = V \Sigma U^T \tag{3.85}$$

3.5. ORTHOGONAL COMPONENT ANALYSIS

As discussed in the previous section, data matrices can be represented in two spaces. Analysis of the characteristics of the data cloud can be done by means of eigenvalues or eigenvectors, if one adapts a least-square-based projection method. The treatment was done from a pure algebraic point of view. In this section, we will include a statistical interpretation of the data matrix and collection of samples of a certain dimension. The least-square minimization then becomes a variance maximization. The "data" are no longer simple algebraic values but are now considered outcomes of some multivariate distribution. The workhorse of multivariate statistical analysis is principal component analysis (PCA), which follows immediately from the previous section. It lies at the foundation of other types of data analysis, certainly those that deal with orthogonal axis systems, least squares, covariances, linear regressions, and anything within that realm.

3.5.1. Principal Component Analysis

3.5.1.1. Theory. We now consider the realizations in the data matrix to be realizations of a random vector of dimension N. Each entry in this random vector $\mathbf{X}_N$ is a random variable X_n, or

$$\mathbf{X}_N = (X_1, X_2, \ldots, X_N)^T \tag{3.86}$$

of which we have L samples/realizations

$$\mathbf{x}_N^{(\ell)} = \left(x_1^{(\ell)}, x_2^{(\ell)}, \ldots, x_N^{(\ell)}\right)^T \quad \ell = 1, \ldots, L \tag{3.87}$$

From a statistical point of view, dimension reduction can be achieved by creating standardized linear combinations (SLC of the elements in random vectors):

$$\mathrm{SLC} = \mathbf{u}^T \mathbf{X}_N = \sum_{n=1}^{N} u_n X_n \quad \text{with} \quad \sum_{n=1}^{N} u_n^2 = 1 \tag{3.88}$$

The goal is to simplify the relationships between the various X_n, such that the newly obtained random vector contains random variables with simpler and easier representation of statistical properties; for example, a transformation that minimizes the correlation between the components in Eq. (3.86). In Section 3.4.2 we used a least-square criterion to minimize the loss of information. Here we treat the same problem differently, namely we would like this new random vector to be as close as possible in variance to the original random variable. Information is now as expressed as a variance (note that a variance is least square (!) deviation from mean, so a comparison with two-norms in Eq. (3.73) is not coincidental as it looks).

Practically, we maximize

$$\mathrm{Var}\left(\mathbf{u}^T \mathbf{X}_N\right) = \mathbf{u}^T \mathrm{Var}(\mathbf{X}_N) \mathbf{u} = \mathbf{u}^T C_N \mathbf{u} \tag{3.89}$$

Here C_N is the covariance matrix of the N random variables and has size $N \times N$. Hence, we need to solve

$$\max_{\mathbf{u}} \ \mathbf{u}^T C_N \mathbf{u} \ \text{ subject to } \ \|\mathbf{u}\| = 1 \tag{3.90}$$

This problem is similar to the algebraic projection problem in $\mathbb{R}^N$ as outlined in Section 3.4.3, where the matrix to be decomposed was $X^T X$, which basically serves as the basis for calculating the (empirical) covariance matrix (although in algebra such interpretation is not given). Therefore, the solution to Eq. (3.90) is found by calculating the eigenvalues and eigenvectors of the covariance matrix, which can be estimated from the data (see next section). Note that this covariance matrix may be extremely large.

Without yet considering any practical calculation, we write the probabilistic result of this "principal component" transformation as

$$\mathbf{Y}_N = (Y_1, Y_2, \ldots, Y_N)^T = U^T (\mathbf{X}_N - E[\mathbf{X}_N]) \tag{3.91}$$

where we centered the original random variable. By noting that V contains the eigenvectors of C_N with eigenvalues λ_n, $n = 1, \ldots, N$, we have the following properties for $\mathbf{Y}_N$:

$$\begin{aligned} &E[Y_n] = 0 && n = 1, \ldots, N \\ &\mathrm{Cov}[Y_n, Y_{n'}] = 0 && n, n' = 1, \ldots, N \;\; n \neq n' \\ &\mathrm{Var}[Y_n] = \lambda_n && n = 1, \ldots, N \\ &\mathrm{Var}[Y_1] \geq \mathrm{Var}[Y_2] \geq \cdots \geq \mathrm{Var}[Y_N] \geq 0 \end{aligned} \tag{3.92}$$

In other words, we obtain a new random vector, whose mean is zero, whose components are not correlated, and whose variances are ranked from high to low, and given by the eigenvalues of the covariance of the original random vector.

3.5.1.2. Practice. The above formulation relies on the theoretical covariance matrix, as defined by expectations. In practice, all the expectations need to be substituted with empirical estimates, subject to their own variances/errors. Consider the estimates of mean and covariance as

$$\begin{aligned} &E[\mathbf{X}_N] \to \bar{\mathbf{x}}_N \\ &C_N \to S_N \end{aligned} \tag{3.93}$$

Then using the eigenvalue decomposition on S_N: $S_N = U^T \Lambda_S U$, we obtain the principal components as

$$\mathbf{y}_N^{(\ell)} = U^T \left(\mathbf{x}_N^{(\ell)} - \mathbf{1}_N \bar{\mathbf{x}}_N^T \right) \tag{3.94}$$

With $\Lambda_S = \mathrm{diag}(\lambda_{S,1}, \ldots, \lambda_{S,N})$, the following property follows

$$s_{y_n}^2 = \lambda_{S,n} \; n = 1, \ldots, N \tag{3.95}$$

meaning that the empirical variances $s_{y_n}^2$ depend on the eigenvalue of S_N, also the empirical covariances are zero. Because covariances and variances depend on the scale of a variable, PCA is sensitive to scale changes, such as simply multiplying a variable with a constant. To mitigate this effect, one will need to scale the variables to the same or similar scale. In summary,

$$\begin{aligned} &\text{Calculate } \bar{\mathbf{x}}_N = \frac{1}{L} \sum_{\ell=1}^{L} \mathbf{x}_N^{(\ell)} \\ &\text{Center } S_N = \frac{1}{L} \sum_{\ell=1}^{L} \left(\mathbf{x}_N^{(\ell)} - \bar{\mathbf{x}}_N \right) \left(\mathbf{x}_N^{(\ell)} - \bar{\mathbf{x}}_N \right)^T \\ &\text{Decompose } S_N = U^T \Lambda_S U \\ &\text{Project } \mathbf{y}_N^{(\ell)} = U^T \left(\mathbf{x}_N^{(\ell)} - \mathbf{1}_N \bar{\mathbf{x}}_N^T \right), \; \ell = 1, \ldots, L \end{aligned} \tag{3.96}$$

PCA is not just a "trick" to orthogonalize data and look for combinations of maximum variance. The resulting linear combination and variances contain interesting information that need to be interpreted. Since PCA relies on a linear combination of components of a random vector, it is imperative to look at the resulting weights. The weighting informs which directions best explain variance; hence, it is useful to plot a measure of how well the first n' components explain that variance. This proportion is given as the ratio:

$$\frac{\sum_{n=1}^{n'} \lambda_{S,n}}{\sum_{n=1}^{N} \lambda_{S,n}} = \frac{\sum_{n=1}^{n'} s_{y_n}^2}{\sum_{n=1}^{N} s_{y_n}^2} \tag{3.97}$$

To summarize this information for all $n' = 1, \ldots, N$, a so-called scree plot is created (either as individual contribution or as cumulative contribution). It allows for a direct visual inspection of how much variance the first n' components explain. Then for some given desired variance, one can read off the corresponding n'.

3.5.1.3. Application of PCA to Spatial Models. To illustrate PCA and its relevance to UQ, consider a case where the matrix X contains $L = 1000$ models of some spatial variable (e.g., porosity, hydraulic conductive, saturation). The dimension of a model is 125×100, hence $N = 12{,}500$. Figure 3.11 shows a few models; the models appear to exhibit smooth spatial variability. Figure 3.12 shows the

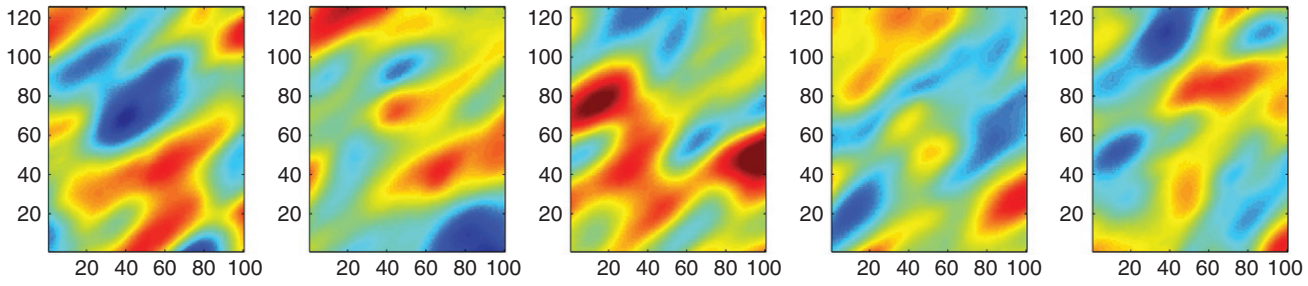


Figure 3.11 Five spatial models out of a set of 1000.

resulting PCA. First, we calculate the cumulative contribution of each principal component (PC) to the variance and create the scree plot. From this plot, we can deduce that 34 PCs are required to explain 95% of variance. Although the size of the covariance matrix is $N \times N$, only a maximum of $L-1$ eigenvalues exist, because of the limited amount of models used.

We can also create a score plot, here the first versus second score (first two entries in $\mathbf{y}_N$), as axes label (and this a convention throughout the book), we list the percent contribution to the variance, 10 and 8% respectively in Figure 3.12. In the score plot, each dot represents an image. The scatter appears to be a bivariate Gaussian distribution with zero correlation. The mean of the realizations is also shown and is close to zero almost everywhere. Next we plot the PCs, the vectors $\mathbf{u}$ of U. Since X contains "images," the resulting PCs are images as well. They also appear to have a meaning: the first PC seems to contain a spatial basis function that is elliptical, somewhat centered in the middle. The second PC seems to contain two ellipses (red and blue areas), one positive and one negative; hence, this represents some gradient in the image. The next PCs contain more ellipses with increasing spatial frequency. Why is this? The mathematical reason for this will be studied in Section 3.7.5. The spatial models here are realizations of a Gaussian process (Section 3.7.5), which requires specifying a mean and spatial covariance. It seems that any realization of such process can be written as a combination (with standard Gaussian coefficients) of filters or basis functions (the

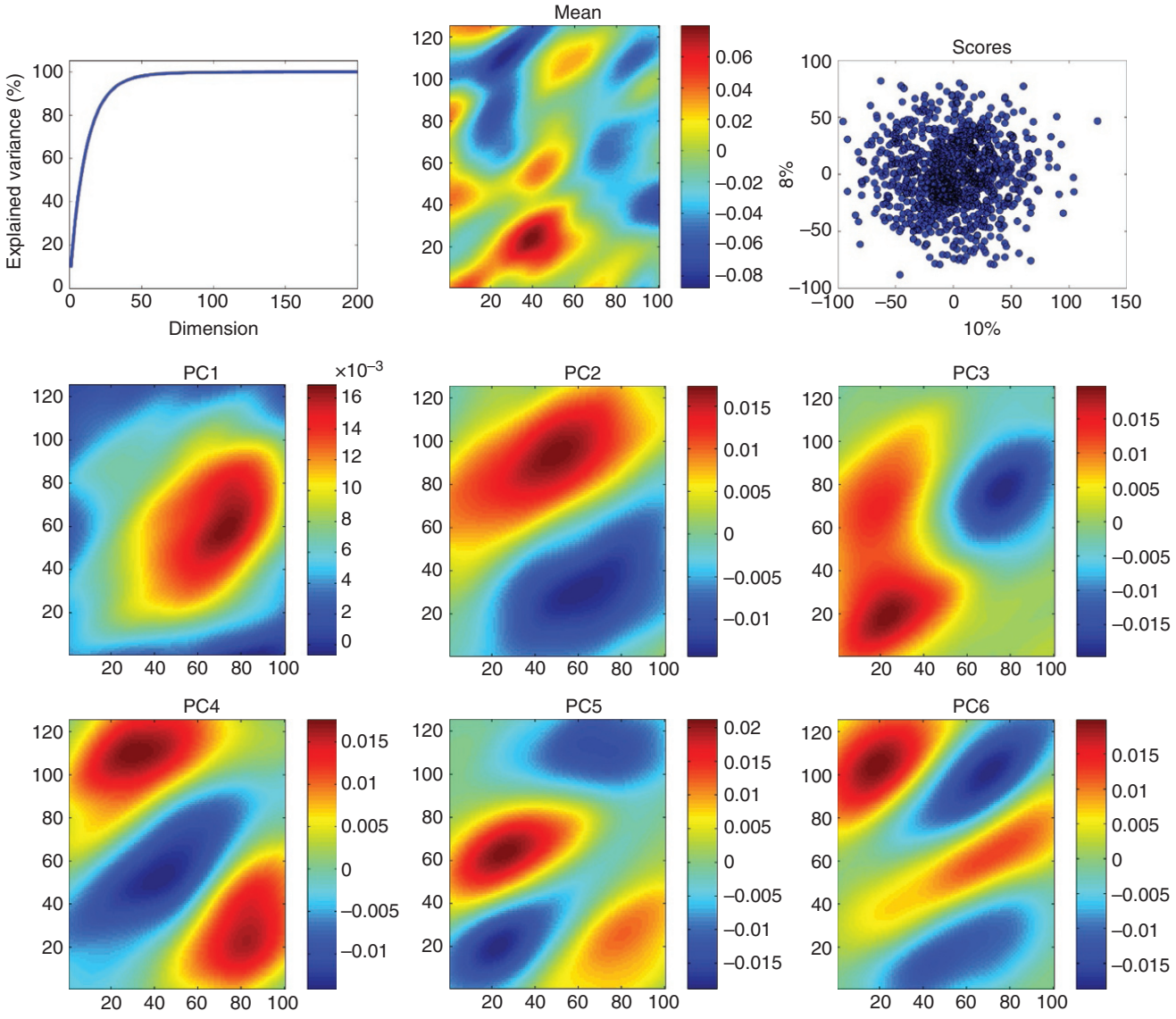


Figure 3.12 Scree plot, average of all models, score plot and 6 out of 1000 PCs, each of dimension 125 × 100.

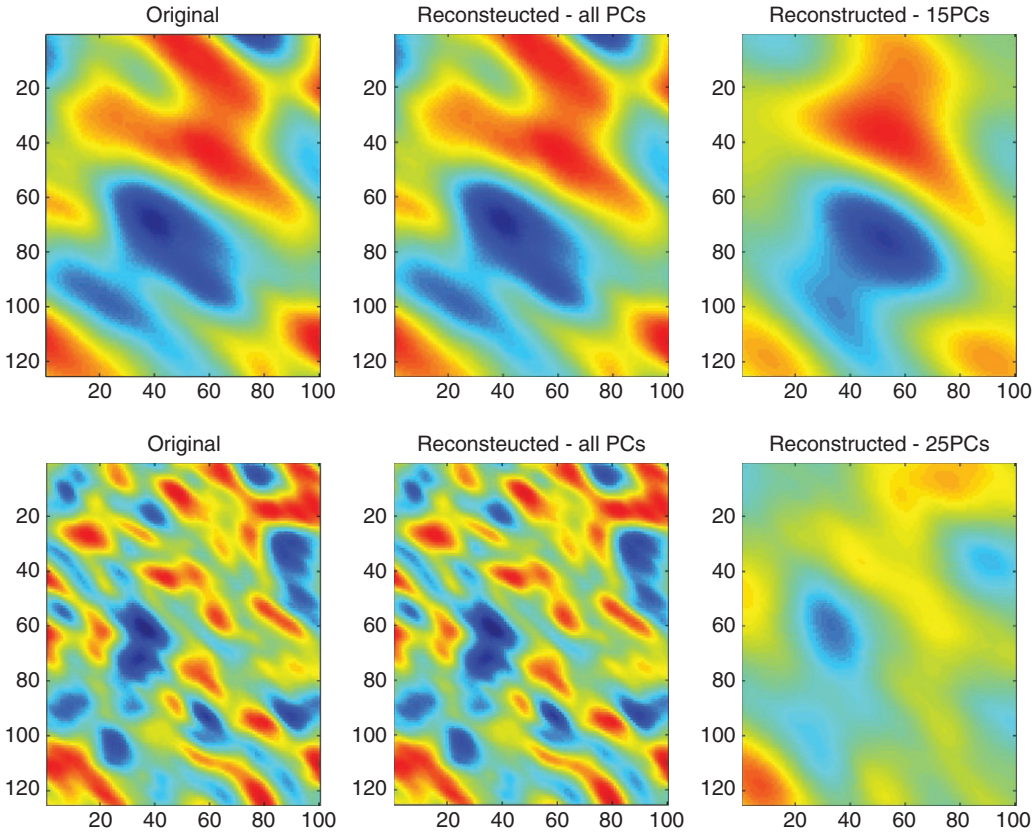


Figure 3.13 Examples of reconstruction of two spatial models with different spatial variability.

PCs) with different spatial frequencies. Hence, it provides a decomposition of the images into a basis and a set of random coefficients.

Figure 3.13 illustrates the bijective nature of PCA, meaning that an image can be reconstructed back, once all PCs and scores are used. By using an amount of PC less than L, we find only an approximation of the original images:

$$\mathbf{x}^{(\ell)}_{N,\text{reconstructed}} = \sum_{n=1}^{n'} \left[\mathbf{y}^{(\ell)}_N\right]_n \mathbf{u}_n + \mathbf{1}_N \bar{\mathbf{x}}^T_N \tag{3.98}$$

Even when using a limited amount of PCs ($n' = 15$) we find a reasonable approximation of the original image $\mathbf{x}^{(\ell)}_N$. This is less so when the spatial variability becomes less smooth, see the bottom row of Figure 3.13. This makes intuitive sense: higher-frequency models require more frequencies (PCs) to represent them.

Decomposition and reconstruction with PCA is appealing because it is bijective. However, the method does not work so well for variables that are not Gaussian, or variables that have physical bounds, such as the concentrations. Figure 3.14 illustrates this, where PCA is applied to a set of concentration curves. PCA reveals that only a few PCs are required to explain variance. This makes sense, the variation in these curves is not complex; one could easily fit a parametric function with a few parameters. However, the PCs, which are unbounded, have negative values as shown in Figure 3.14. As a result, any reconstruction with limited PCs will lead to unphysical values (negative concentrations).

3.5.2. Multidimensional Scaling

3.5.2.1. Basic Example. In Section 3.4.4, we presented a dual algebraic view on the data matrix, one based on X^TX and one based on XX^T. The statistical variation of the same type of orthogonal analysis, termed PCA is based on X^TX which is essentially the empirical covariance matrix (up to a factor of $1/L$). It therefore makes sense when $L \ll N$ to look at the dual of PCA, which will now be based on the dot-product XX^T. This method is termed multidimensional scaling (MDS).

In presenting MDS, we will not start from a Cartesian axis system and space, but we will start from a space in which only distances are defined: a metric space. As an illustrative example, consider L cities and a distance

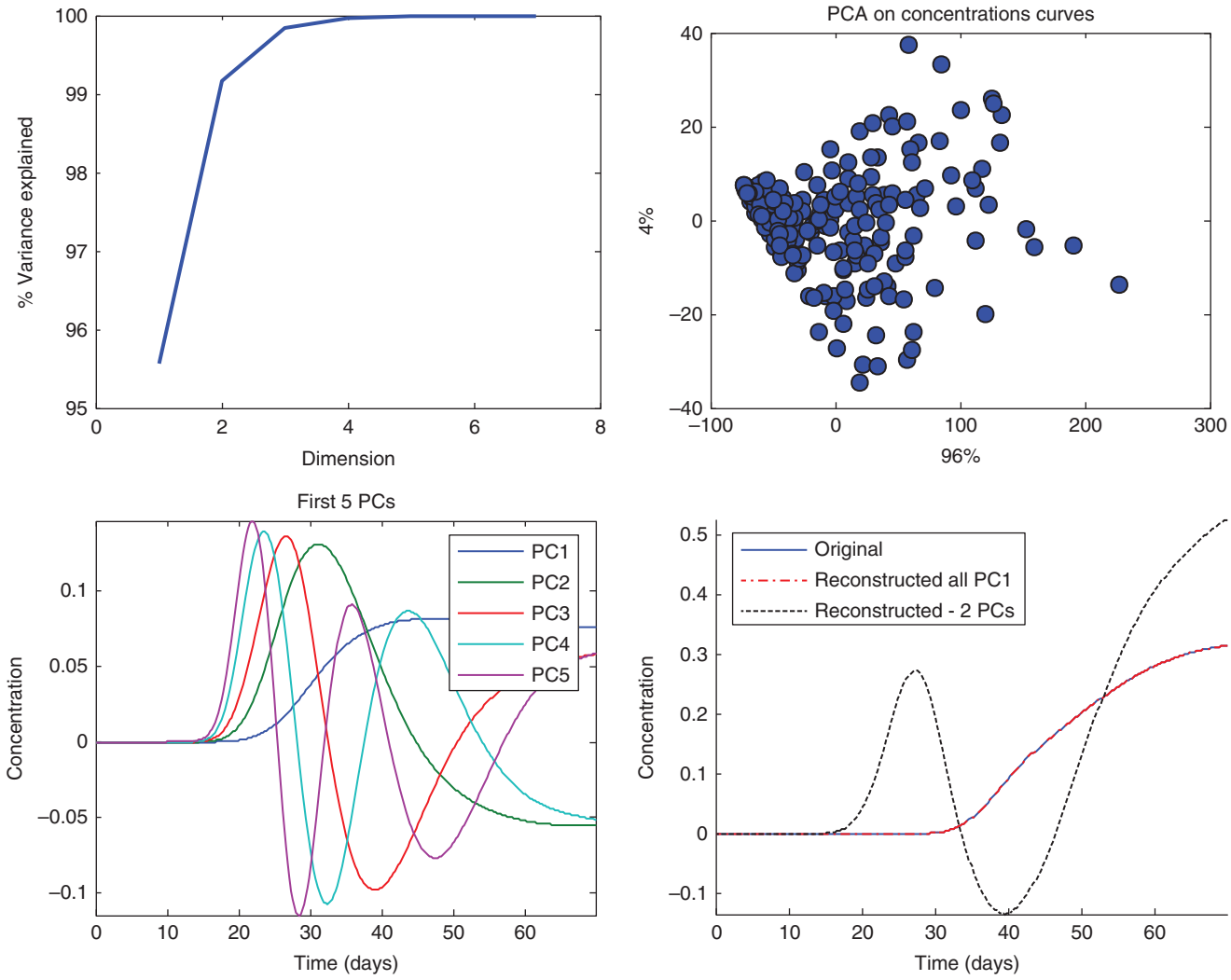


Figure 3.14 PCA applied to the set of concentration curve of Figure 3.2.

table between those cities (see Figure 3.15). Like PCA, MDS is a projection method, but now such projection occurs from a metric space into a Cartesian space. More specifically, MDS finds the projection that best matches the distance in the distance table. Like PCA, such projection can be done in $1D$, $2D$, …, LD. Note that the maximal projected Cartesian space has dimension L (and not N as for PCA). In that sense, it is also a dimension reduction method.

3.5.2.2. MDS Projection. In MDS, we start from an $L \times L$ distance matrix. This could be the differences between any two models, any two data responses, or any two predictions. What is important is that from now on we will work with $\mathbf{x}_N^{(\ell)}, \ell = 1, \ldots, L$ (column-wise) and not $\mathbf{x}_n^{(L)}, n = 1, \ldots, N$ (row-wise) as in PCA. To lighten the notation, we will write $\mathbf{x}^{(\ell)}$ instead of $\mathbf{x}_N^{(\ell)}$. Consider first the Euclidean distance:

$$d_{\ell\ell'}^2 = \left(\mathbf{x}^{(\ell)} - \mathbf{x}^{(\ell')}\right)^T \left(\mathbf{x}^{(\ell)} - \mathbf{x}^{(\ell')}\right), \; 1 \le \ell, \ell' \le L \;\; \mathbf{x}^{(\ell)}, \mathbf{x}^{(\ell')} \in \mathbb{R}^N \tag{3.99}$$

The usual notation of $d_{ij}^2 = \left(\mathbf{x}_i - \mathbf{x}_j\right)^T \left(\mathbf{x}_i - \mathbf{x}_j\right)$ is now put in the specific context of UQ problems. The aim of MDS is to recover in lower dimensions, the original Cartesian coordinates of $\mathbf{x}^{(\ell)}$ from a distance matrix, in this case containing Euclidean distances. However, since our dual treatment of the data matrix only works on either the covariance or the dot-product matrices, we need to retrieve the dot-product matrix from the Euclidean

	Boston	NYC	DC	Miami	Chic	Seattle	SF	LA	Denver
Boston	0	206	429	1504	963	2976	3095	2979	1949
NYC	206	0	233	1308	802	2815	2934	2786	1771
DC	429	233	0	1075	671	2684	2799	2631	1616
Miami	1504	1308	1075	0	1329	3273	3053	2687	2037
Chic	963	802	671	1329	0	2013	2142	2054	996
Seattle	2976	2815	2684	3273	2013	0	808	1131	1307
SF	3095	2934	2799	3063	2142	808	0	379	1235
LA	2979	2786	2631	2687	2054	1131	379	0	1059
Denver	1949	1771	1616	2037	996	1307	1235	1059	0

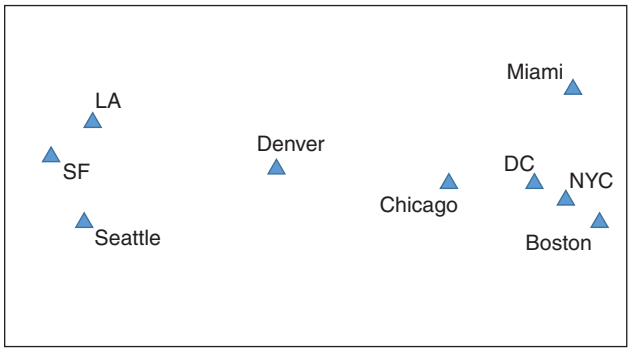


Figure 3.15 MDS: turning a distance table between cities into a 2D map.

distance matrix. To achieve this, we rely on the following relationship:

$$\begin{aligned} d^2_{\ell\ell'} &= \mathbf{x}^{(\ell)^T}\mathbf{x}^{(\ell)} + \mathbf{x}^{(\ell')^T}\mathbf{x}^{(\ell')} - 2\mathbf{x}^{(\ell)^T}\mathbf{x}^{(\ell')} \\ &= b_{\ell\ell} + b_{\ell'\ell'} - 2b_{\ell\ell'} \end{aligned} \tag{3.100}$$

with $b_{\ell\ell'}$ a dot-product between $\mathbf{x}^{(\ell)}$ and $\mathbf{x}^{(\ell')}$. Similar to centering operations on the covariance matrix in PCA, we need to center the dot-product matrix to obtain

$$\sum_{\ell=1}^{L} b_{\ell\ell'} = 0 \tag{3.101}$$

This is not as trivial because we can no longer calculate a mean directly from data. Instead, we impose this constraint as follows:

$$\begin{aligned} \frac{1}{L}\sum_{\ell=1}^{L} d^2_{\ell\ell'} &= \frac{1}{L}\sum_{\ell=1}^{L} b_{\ell\ell} + b_{\ell'\ell'} \\ \frac{1}{L}\sum_{\ell'=1}^{L} d^2_{\ell\ell'} &= b_{\ell\ell} + \frac{1}{L}\sum_{\ell=1}^{L} b_{\ell'\ell'} \\ \frac{1}{L^2}\sum_{\ell'=1}^{L}\sum_{\ell=1}^{L} d^2_{\ell\ell'} &= \frac{2}{L}\sum_{\ell=1}^{L} b_{\ell\ell} \end{aligned} \tag{3.102}$$

With this centering, we can now write the dot-product as a function of the Euclidean distance:

$$b_{\ell'\ell'} = -\frac{1}{2}d^2_{\ell\ell'} - \frac{1}{2}\left(d^2_{\ell\bullet} + d^2_{\bullet\ell'} - d^2_{\bullet\bullet}\right) \tag{3.103}$$

where the $\bullet$ refers to the summing over the indices Eq. (3.103). We can also put the same expression in matrix notation. Consider, then, A the matrix containing the entries $-\frac{1}{2}d^2_{\ell\ell'}$ and H the centering matrix, in this case, $= I_L - \frac{1}{L}\mathbf{1}_L\mathbf{1}_L^T$. Then the dot-product matrix B can be written as

$$\begin{aligned} B &= (HA)(HA)^T \\ &= XX^T \end{aligned} \tag{3.104}$$

Now consider the eigenvalue decomposition of B

$$\begin{aligned} B &= V_B\Lambda_B V_B^T \\ &= \left(V_B\Lambda_B^{1/2}\right)\left(V_B\Lambda_B^{1/2}\right)^T \end{aligned} \tag{3.105}$$

Hence, the coordinates of X can be reconstructed by means of

$$X = V_B\Lambda_B^{1/2} \tag{3.106}$$

One of the strengths of MDS is that any distance can be used, not just the Euclidean distance. Figure 3.16 illustrates overhead images of 136 from a flume experiment (see Chapter 5 for details on this data set). The modified Hausdorff distance is calculated for each pair of images. MDS is then used to create a 2D map of these images (just like mapping of cities). In this map, images that look alike (small distance) are plotted close by.

3.5.2.3. Kernel Density Estimation in Metric Space. MDS generates a Euclidean space of dimension $n \ll L$. The higher the dimension the better that space approximates the variability of the physical variable as modeled by the distance defined. The Euclidean space approximates how close variables are with respect to each other; hence, if the original variable is a stochastic variable (of very high dimension possibly), then MDS creates an empirical density of these variables in lower-dimensional space. If the approximation can be done in a low dimension (e.g., <6), then one can estimate the pdf from the cloud of points. A useful method in that regard is kernel density estimation [*Silverman*, 1986, see Section 3.3.2]. When applied to a set of reconstructed coordinates $\mathbf{x}_n^{(\ell)}, \ell = 1,\ldots,L$ with some low dimension $n \ll L$, then a kernel density estimate is

$$f(\mathbf{x}_n) = \frac{1}{L}\sum_{\ell=1}^{L} K\left(\frac{\mathbf{x}_n - \mathbf{x}_n^{(\ell)}}{\sigma}\right) \tag{3.107}$$

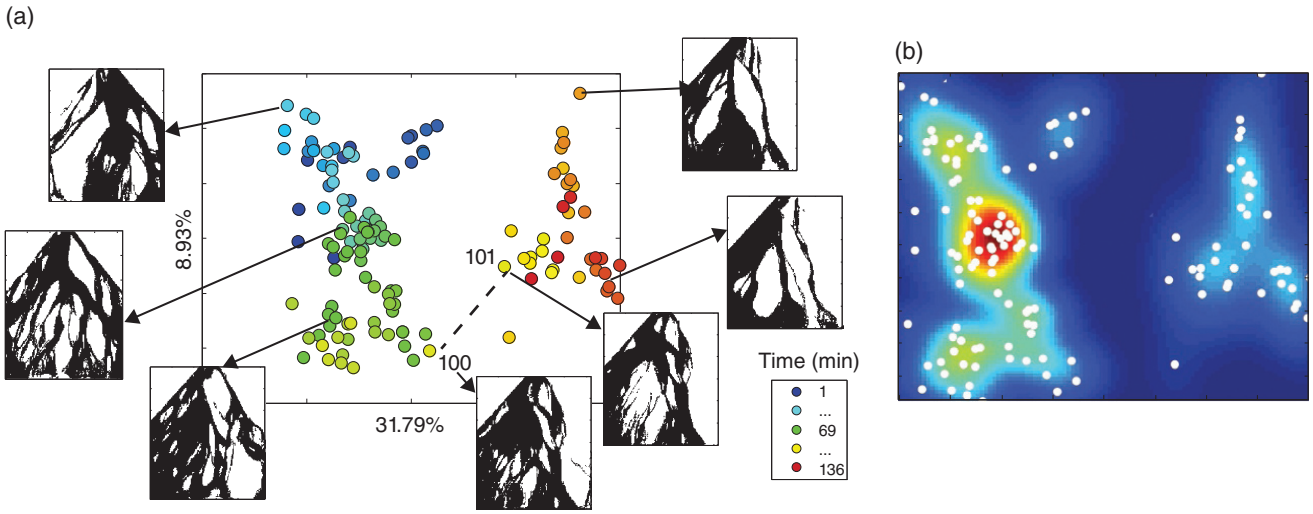


Figure 3.16 (a) MDS projection of 136 binary overhead snapshots of a flume experiment, using the modified Hausdorff distance. (b) Kernel density estimation in MDS projection.

Figure 3.16 shows an example of kernel density estimation applied to the flume experiment images.

As another example, consider again the simple hydro case. First, we calculate the Euclidean difference between any two model realizations in our hydro case:

$$d^2_{\ell\ell'} = \left(\mathbf{m}^{(\ell)} - \mathbf{m}^{(\ell')}\right)^T \left(\mathbf{m}^{(\ell)} - \mathbf{m}^{(\ell')}\right), \quad 1 \le \ell, \ell' \le L \tag{3.108}$$

or simply the square difference between values in the grid. Note that these realizations have different variogram ranges, nuggets, and types. Figure 3.17 shows the score plot colored by the type of variogram (spherical vs. Gaussian).

Consider now calculating the distance between the concentration responses calculated from these hydraulic conductivity (Figure 3.17). The score plot of Figure 3.17 looks a lot like the score plot of Figure 3.14. This makes sense, because of the duality of MDS with Euclidean distance and PCA. An interesting observation in Figure 3.17 is the coloring of the dots with parameter values that were used to generate the models. In the first plot, we observe that the red and blue dots occur randomly, while in the second plot (mean k), we observe a clear trend. The interpretation is that mean k impacts the response while the variogram does not. This will be used in Chapter 4 to develop methods of sensitivity analysis.

3.5.3. Canonical Correlation Analysis

Continuing our study of "data" and the analysis of data matrices from an algebraic and statistical point of view, we now study two data matrices jointly and how they relate to each other. We will again rely on a least-squares framework with Cartesian axis, Euclidean distances, and orthogonal projections. A method of multivariate analysis for exploration and quantification of the relationships between two data matrices under these principles is termed canonical correlation analysis (CCA). Like PCA and MDS, CCA is an orthogonal method that relies on projections in lower dimensions.

3.5.3.1. Theory. Consider two random vectors $\mathbf{X}$, $\mathbf{Y} \in \mathbb{R}^N$ (or $\mathbb{R}^L$, or they can be of different dimension). We consider the linear combinations

$$\begin{aligned} &\mathbf{a}^T\mathbf{X} \\ &\mathbf{b}^T\mathbf{Y} \end{aligned} \tag{3.109}$$

The correlation between these two random variables can be derived as

$$\begin{aligned} \rho\left(\mathbf{a}^T\mathbf{X}, \mathbf{b}^T\mathbf{Y}\right) &= \frac{\mathbf{a}^T \operatorname{cov}(\mathbf{X},\mathbf{Y})\mathbf{b}}{\sqrt{\mathbf{a}^T \operatorname{var}(\mathbf{X})\mathbf{a}}\sqrt{\mathbf{b}^T \operatorname{var}(\mathbf{Y})\mathbf{b}}} \\ &= \frac{\mathbf{a}^T C_{XY}\mathbf{b}}{\sqrt{\mathbf{a}^T C_{XX}\mathbf{a}}\sqrt{\mathbf{b}^T C_{YY}\mathbf{b}}} \end{aligned} \tag{3.110}$$

In CCA we find the maximum of this correlation. This maximum depends on the singular value decomposition (eigenvalues in case $\dim(\mathbf{X}) = \dim(\mathbf{Y})$) of the following matrix

$$C = C_{XX}^{1/2} C_{XY} C_{XX}^{1/2} = V \Lambda U^T \tag{3.111}$$

namely, the linear combinations that result in maximal correlation are the canonical correlation vectors

$$\begin{aligned} \mathbf{a}_k^c &= C_{XX}^{-1/2}\mathbf{v}_k \qquad k = 1, \ldots, \operatorname{rank}(C) \\ \mathbf{b}_k^c &= C_{YY}^{-1/2}\mathbf{u}_k \end{aligned} \tag{3.112}$$

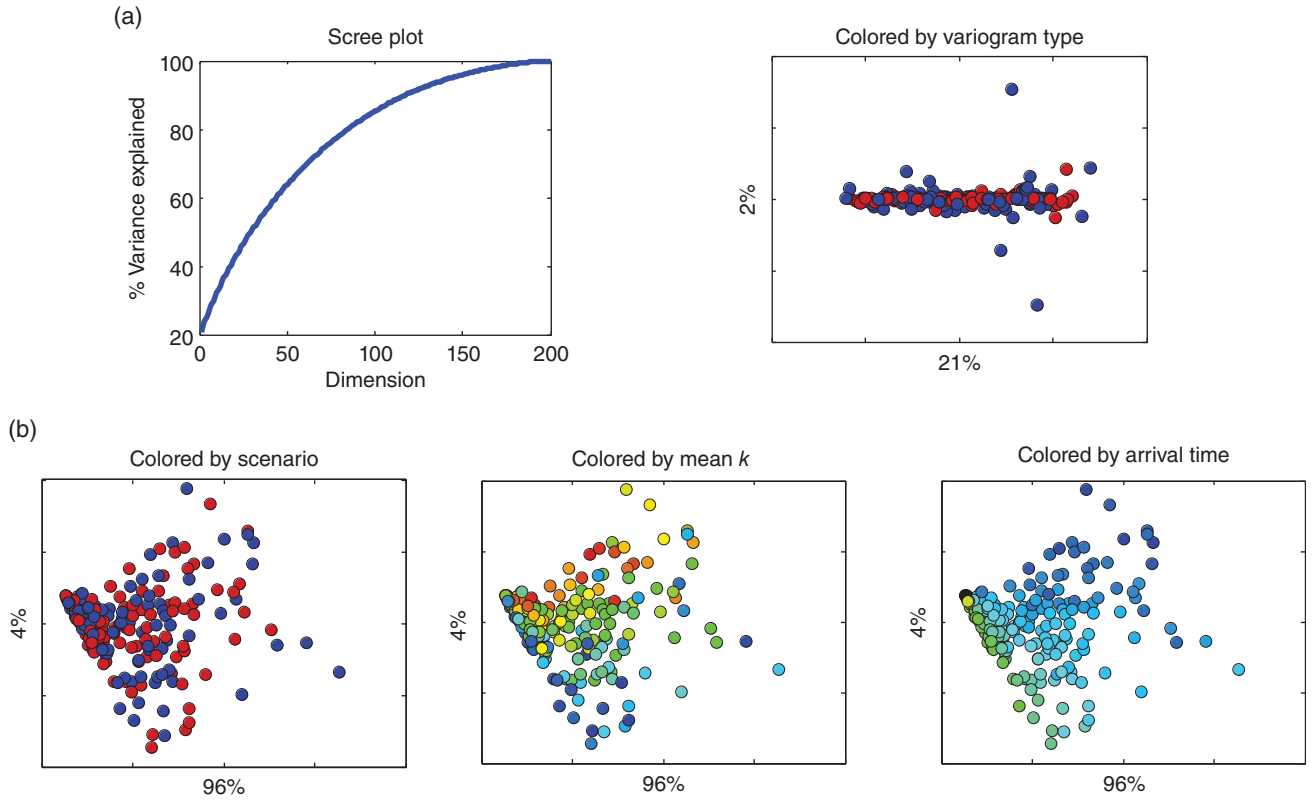


Figure 3.17 (a) Scree plot and MDS projection when the distance is the Euclidian distance between hydraulic conductivity realizations. (b) MDS projection when the distance is the Euclidian distance between realizations of concentration in time. The plots are colored by model parameters.

This results in a set of new vectors whose pair-wise components are maximally correlated (and hence cross-components are minimally correlated):

$$X_k^c = \left(\mathbf{a}_k^c\right)^T \mathbf{X} \qquad k = 1, \ldots, \text{rank}(C)$$
$$Y_k^c = \left(\mathbf{b}_k^c\right)^T \mathbf{Y} \tag{3.113}$$

The maximal correlations are function of the eigenvalues

$$\rho_{\max,\mathrm{k}}^c = \sqrt{\lambda_k} \tag{3.114}$$

3.5.3.2. Example. In constraining predictions, we often use data. For example, we use pressure data to constrain permeability or head data to constrain hydraulic conductivity. The hope is that by building calibrated models, the prediction will have less uncertainty than not doing so. This kind of calibration (inversion, Chapter 6) becomes very difficult when models and data become very complex and high dimensional. In Chapter 7, we will develop an alternative approach that "learns" prediction directly from data. In such learning CCA will be used as one method to learn. A simple example is as follows. Consider the models generated in our hydro case and consider that both data variables and prediction variables have been evaluated. The data consists of seven measurements of hydraulic head, the prediction is the concentration. They should be related. The question is how this can be visualized, after all we are dealing with a seven-dimensional variable and a function. To do this, we apply first PCA on each variable (see Figure 3.18). The first PC of the data does not correlate with the first PC of the prediction. CCA finds the optimal linear combinations that find this correlation. Consider that for the data we retain two PCs (about 90%) and for the prediction we retain five PCs. This means that a maximum of two canonical components can be calculated, as shown in Figure 3.18. We note the high correlation of 0.86 in the first components, which is encouraging since it means that data are correlated with the prediction, so reduction in uncertainty on the prediction is possible. In Chapter 7, we will use CCA to build linear relationship between complex objects, whether vectors, functions, or 2D/3D maps.

3.6. FUNCTIONAL DATA ANALYSIS

3.6.1. Introduction

Multivariate data analysis comprises methodologies to address the statistical analysis of data and inference with

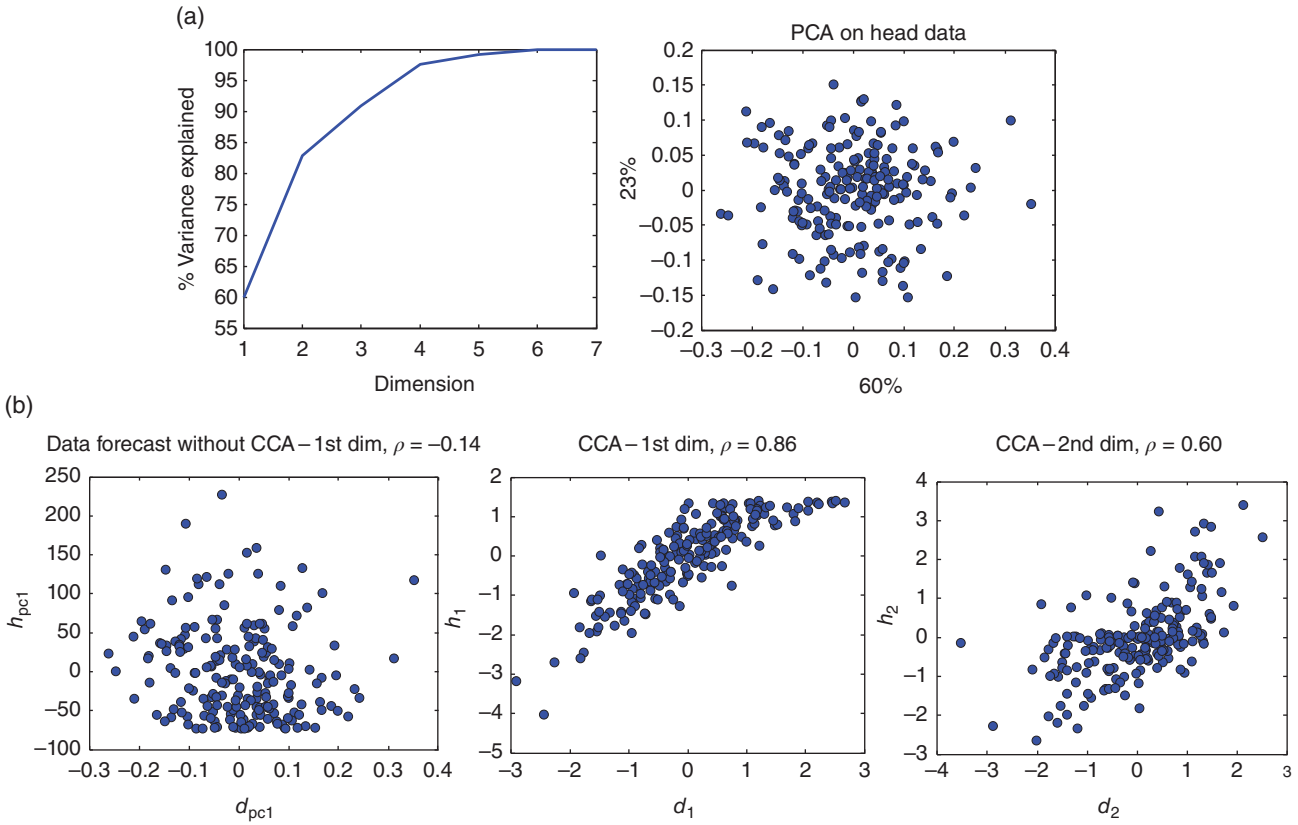


Figure 3.18 (a) PCA of seven head measurements and score plot. (b) Correlation between PCs is almost zero. CCA shows correlation of 0.86 and 0.6 between the canonical components of data and prediction.

high-dimensional data and models. In certain cases, it is however more effective to resort to functional data analysis (FDA) [*Ramsay and Silverman*, 2005]. The decision between the two is a subjective choice. Broadly speaking, FDA relies on the notion that the data has some systematic variation. In the context of this book, this can refer to cases where physical dynamics are driving the system, but because of uncertainty, the exact nature is not fully known; only what that systematic part looks like. For example, a concentration of some substance starts at zero and then gradually increases to level off at 100%, but we do not know when that starts or how fast or when it levels off. In the latter case, the variable is a function of time, and hence in theory, infinite-dimensional. In multivariate analysis, we would instead discretize time into small intervals and treat time instance variables as high-dimensional (and highly correlated). This is not required in FDA and is a major advantage. FDA is a non-parametric approach. It does not rely on fitting functions that represent physics. For example, in the production of shale gas (see Figure 3.19) the function is simply a peak (peak gas) followed by some decline. The parametric approach would be to fit these parametric functions to observed decline data to get parameter estimates. The problem is that the functional form has to be known and currently it is not clear, for shale gas production, which forms are appropriate. FDA avoids this by using standard basis functions that can be used in many applications.

3.6.2. A Functional Basis

More formally, FDA assumes that changes in any measurement of a physical variable over space or time is based on an underlying smooth physical process that in turn can be mathematically represented using a continuous and differentiable mathematical function and that this function not always needs to be known for analyzing measurements. This assumption allows for the decomposition of any time series measurement into a linear combination of underlying continuous functions called basis functions, forming a functional basis. Multiple functional bases such as a sinusoidal basis, a Fourier basis, a polynomial basis, an exponential basis, or a spline basis are available and the choice between them is application driven. The spline basis has an advantage over the others because of its versatility in terms of computational ease

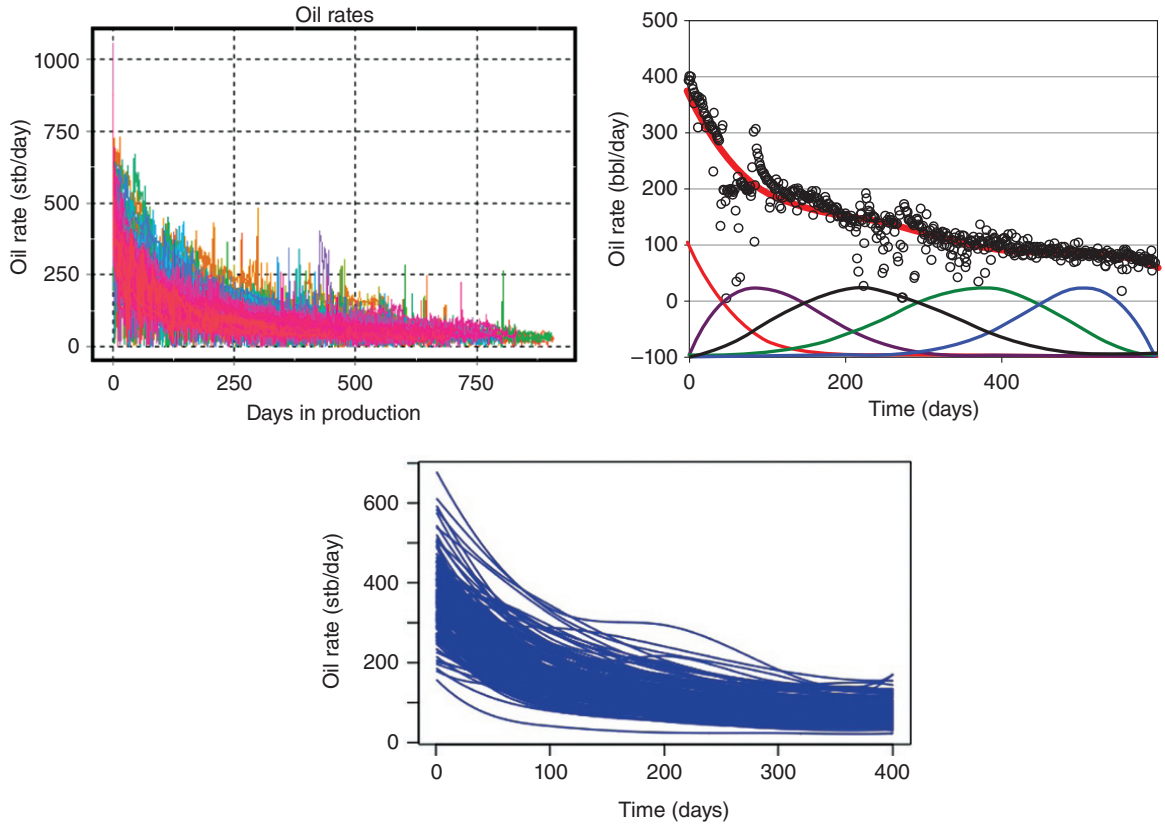


Figure 3.19 Raw data of oil rate decline in 200 wells; basis function and fitting with smoothing; all 200 smoothed data.

of evaluation, as well as their derivatives. This flexibility will be used throughout this book. Consider a time series $x(t)$ as an example. Using a spline basis of K spline functions $\{\psi_1(t), \psi_2(t), \ldots, \psi_K(t)\}$, the time series is approximated by a linear combination

$$x(t) \cong \sum_{k=1}^{K} c_{\psi,k}\psi_k(t) \tag{3.115}$$

where the FDA components $c_{\psi,k}$ are the scalar linear combination coefficients of the spline function $\psi_k(t)$. In terms of matrix notation, consider matrix X containing L samples of time series sampled at N time instances, the FDA composition can be written as

$$X \simeq C^{\psi}\Psi \tag{3.116}$$

where the $K \times N$ matrix Ψ contains the values of the K spline basis functions at the N values of time $t = t_1, t_2, \ldots, t_N$ as row vectors, and the $L \times K$ matrix C^{ψ} contains the K coefficients or FDA components of each of the L time series as row vectors.

Consider the example of shale oil production from 200 wells as shown in Figure 3.19. The coefficients in Eq. (3.115) are found by least-square fitting with a regularization term

$$\underset{c_{\psi,k}}{\operatorname{argmin}} \sum_{\ell=1}^{L}\left(x(t) - \sum_{k=1}^{K} c_{\psi,k}\psi_k(t)\right)^2 + \lambda \int \frac{d^2}{dt^2}\left(\sum_{k=1}^{K} c_{\psi,k}\psi_k(t)\right)^2 dt \tag{3.117}$$

The regularization term avoids overfitting noisy data by adding a roughness penalty in terms of the second derivative. FDA therefore requires specifying two tuning parameters: λ the amount of smoothing and K the number of basis functions. These are usually obtained by means of cross-validation [*Ramsay and Silverman*, 2005].

3.6.3. Functional PCA

As discussed in Section 3.5.1, PCA is a multivariate analysis procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated

variables. PCA identifies the principal modes of variation from the eigen-vectors of the covariance matrix. Functional PCA (FPCA) simply consists of doing eigenvalue decomposition on the vectors of component data $c_{\psi,k}$ obtained after functional decomposition. Hence FDA turns a functional problem into a vector problem on which classical multivariate techniques apply. The end result is that the function can be approximated by a linear combination of eigen-functions $\phi_m(t)$:

$$x(t) \cong \sum_{m=1}^{M} c_m^f \phi_m(t) \tag{3.118}$$

Conventionally $M \ll K$, so the PCA step of FPCA achieves a dimension reduction. How do we get to (3.118)? Recall that PCA relies on an eigenvalue decomposition of the covariance matrix of a random vector X:

$$C_N \mathbf{u} = \lambda \mathbf{u} \text{ or } C_N = U^T \Lambda U \tag{3.119}$$

The covariance matrix is of size $N \times N$. In a functional space, however, we define the empirical covariance function of a set of sample functions $\{x^{(1)}(t), \ldots, x^{(L)}(t)\}$:

$$c(t_n, t_{n'}) = \frac{1}{L} \sum_{\ell=1}^{L} x^{(\ell)}(t_n) x^{(\ell)}(t_{n'}) \tag{3.120}$$

The eigen-problem is now defined as an integral

$$\int c(t_n, t_{n'}) u(t_{n'}) dt_{n'} = \lambda u(t_n) \tag{3.121}$$

If we define

$$C\mathbf{u} = \int c(\cdot, t_{n'}) u(t_{n'}) dt_{n'} \tag{3.122}$$

which is an integral transform C of the function u. C is now a covariance operator instead of a covariance matrix, hence

$$C\mathbf{u} = \lambda \mathbf{u} \tag{3.123}$$

which looks like PCA but now with functions. One major difference, however, lies in the rank of the covariance matrix versus the integral transform. In PCA, the rank of the empirical covariance matrix is maximally equal with $L - 1$, because $L \ll N$ the dimension of the vector. In functional analysis the function can be sampled infinitely; hence, N can be very large. Practically, we need to retain only a few dimensions in most cases, here denoted as M. Expression (3.118) is obtained by equating the first M eigen-functions $u_m(t)$, $m = 1, \ldots, M$ with $\phi_m(t)$, $m = 1, \ldots, M$ and the weights (the component scores) as

$$c_m^f = \int \phi_m(t) x(t) dt \tag{3.124}$$

Evidently, all integrals are approximated by sums.

Figure 3.20 illustrates the use of FPCA to the shale oil decline curves. Similar to PCA, a score plot of the FPC can be produced. Additionally, one can plot the eigen-functions, which are often termed "harmonics" referring to components of vibration of a string fixed at each end. Here these "vibrations" are the changes of the function around the mean. To visualize this better, one often plots this mean, added and subtracted with some multiple of the eigen-function. This is shown in Figure 3.20. Now one notices how the first functional component (75.3% of variance) represents a variation around the mean, the second functional component (12.6%) has a stationary point around 50 days. This can be attributed to a change in production from flow due to fracturation to flow through the geological medium. The third component has two stationary points, one early and one around 100 days.

3.7. REGRESSION AND CLASSIFICATION

3.7.1. Introduction

Regression and classification methods are important elements of statistical learning [*Hastie et al.*, 2009]. Numerous methods have been developed, from simple linear regression to nonlinear methods such a neural network or deep learning [see e.g., *Bishop*, 1995]. In general, these are denoted as methods for "predictive learning from data." In selecting a suitable method, one will need to account for a variety of criteria. Table 3.2 provides an overview of some popular methods, together with criteria against which each method can be evaluated. Artificial neural networks (ANN) constitute a large family of nonlinear regression models that create a nonlinear mapping between input and output variables. The initial idea of ANN was to solve problems in the same way the human brain would. A support vector machine (SVM) uses supervised learning to create a classifier that separated regions in space by means of hyperplanes. The k-nearest neighbor is a local classification or regression method that uses an amount k of nearest data near a location in input space that needs to be classified and regressed. The result is simply a majority vote (classification) or an average (regression). Kriging accounts for the correlation between predictors and allows to interpolate data. In this section, we will focus on several of these methods and how they apply in the context of UQ.

3.7.2. Multiple Linear Regression

The simplest form of linear regression is multiple linear regression. It is used to model the relationship between one or more data variables $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ and a single continuous prediction variable Y. The fundamental assumption of multiple linear regression is that the

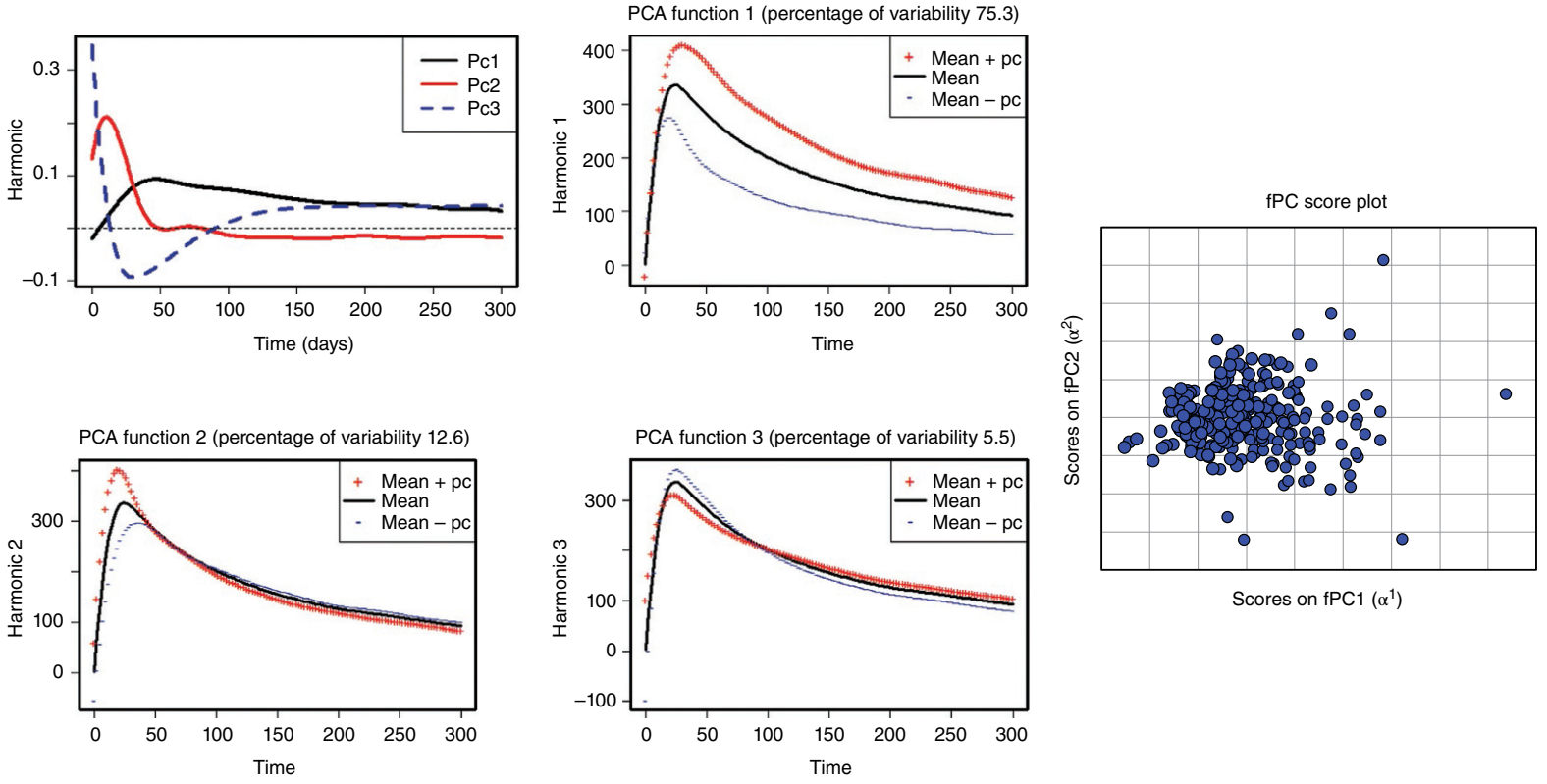


Figure 3.20 FPCA with harmonics, PC functions, and score plot.

Table 3.2 Overview of methods of regression and classification and how they score on various criteria.

Criteria	Neural net	Kriging	SVM	Kernel *k*-NN	Trees	Boosted trees
Mixed-type data	–	+	–	–	+	+
Missing data	–	+	–	+	+	+
Robust to outliers	–	0	–	+	+	+
Computational scalability	–	0	–	–	+	+
Deal with irrelevant input	–	0	–	–	+	+
Ease of interpretation	–	+	–	–	0	–
Predictive power	+	0	+	+	–	+

Source: Idea adapted and extended from *Hastie et al.* [2009].
– = poor; + = good; 0 = average.

conditional expectation function $E(Y|\mathbf{X})$ can be expressed using the linear relation:

$$E(Y|\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N \tag{3.125}$$

β_0 is commonly referred to as the intercept while $\beta_1, \ldots, \beta_n$ are termed coefficients. We also assume the existence of an unobserved random variable ϵ that adds noise to the linear relationship. Given L measurements of $\mathbf{X}$ and Y that are independently and identically distributed, the modeled relationship can be expressed as

$$y^{(\ell)} = \beta_0 + \beta_1 x_1^{(\ell)} + \beta_2 x_2^{(\ell)} + \cdots + \beta_N x_N^{(\ell)} + \epsilon^{(\ell)} \quad \text{for } \ell = 1, \ldots, L \tag{3.126}$$

In matrix notation, this is written as

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.127}$$

where each of the matrices is given as

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_N^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(L)} & \cdots & x_N^{(L)} \end{pmatrix} \tag{3.128}$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_N \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(L)} \end{pmatrix}, \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon^{(1)} \\ \vdots \\ \epsilon^{(L)} \end{pmatrix} \tag{3.129}$$

To solve for the unknown coefficients and intercept, an estimation method is required. A variety of different techniques can be used, for example ordinary least squares (OLS), generalized least squares, ridge regression, maximum likelihood estimation, and so on. Each method varies in terms of computational complexity and underlying assumptions. Refer to *Hastie et al.* [2009] for a discussion of various estimators and their properties. We will consider the simplest and most commonly used estimator: OLS. The solution from OLS in matrix notation is as follows:

$$\hat{\boldsymbol{\beta}} = \left(X^T X\right)^{-1} X^T \mathbf{y} \tag{3.130}$$

Using this estimated $\hat{\boldsymbol{\beta}}$, the predicted values of Y can be computed as $\hat{\mathbf{y}} = X\hat{\boldsymbol{\beta}}$. The difference between these predicted and observed values of Y is called the residual:

$$\hat{\boldsymbol{r}} = \mathbf{y} - \hat{\mathbf{y}} = \left(I - X\left(X^T X\right)^{-1} X^T\right)\mathbf{y} \tag{3.131}$$

The OLS solution is the one that minimizes the sum of the squares of the residuals. It does however, require additional assumptions. First of all, the error random variable should have a zero mean, $E[\boldsymbol{\epsilon}] = 0$, and be uncorrelated with the data variables $E[X^T\boldsymbol{\epsilon}] = 0$. Furthermore, the data variables in X must be linearly independent, the failure of which is termed multicollinearity. Finally, the error term must be homoscedastic, that is, the variance of the errors should not change between different observations. This is expressed mathematically as $E[(\epsilon^{(\ell)})^2|\, X] = \sigma^2$ for $\ell = 1, \ldots, L$, where σ^2 is finite.

When each of the previous conditions are met, OLS can be used to make estimates of the prediction variable for new data observations X_{new}.

$$\hat{\mathbf{y}}_{\text{new}} = X_{\text{new}}\hat{\boldsymbol{\beta}} \tag{3.132}$$

This is also known as the minimum-variance unbiased estimator. By performing this analysis, we can make estimates of a prediction variable using multiple data variables.

3.7.3. Support Vector Machines

SVMs are powerful and popular statistical models used for both regression and classification. It was originally invented as a linear binary classifier [*Vapnik and Lerner*, 1963] but has since seen numerous improvements to

handle nonlinear boundaries, soft margins, regression, and multiple classes. Given a set of training samples, each labeled as being in one of the two classes, the basic SVM aims to train a linear hyperplane that separates the samples according to its class.

The basic SVM is applied when we have a set of real predictor variables X and binary predictor variable $Y \in \{-1, 1\}$. SVM attempts to fit a linear hyperplane $\mathbf{w}^T\mathbf{x} + \mathrm{b} = 0$ that separates the samples into two classes. This means finding the value of $\mathbf{w}$ can then be expressed as an optimization problem:

$$\min_{w} ||\mathbf{w}||$$

subject to

$$y^{(\ell)}\left(\mathbf{w}^T\mathbf{x}^{(\ell)} + \mathrm{b}\right) \geq 1 \quad \forall \ell = 1, \ldots, L \tag{3.133}$$

This can be solved using quadratic programming. Once the hyperplane has been estimated, predictions for a new sample $\mathbf{x}^*$ can be made by evaluating

$$f(\mathbf{x}^*) = \mathrm{sgn}\left(\mathbf{w}^T\mathbf{x}^* + \mathrm{b}\right) \tag{3.134}$$

This optimization problem can be solved using Lagrangian multipliers, in which a multiplier $\lambda^{(\ell)}$ is assigned to each of the constraints. The resulting decision boundary can then be expressed as

$$f(\mathbf{x}^*) = \mathrm{sgn}\left(\sum_{\ell=1}^{L} \lambda^{(\ell)} y^{(\ell)} K\left(\mathbf{x}^*, \mathbf{x}^{(\ell)}\right) + \mathrm{b}\right) \tag{3.135}$$

The function $K(\mathbf{x}^*, \mathbf{x}^{(\ell)}) = \phi(\mathbf{x}^*)^T\phi(\mathbf{x}^{(\ell)})$ is known as the kernel function. ϕ is a function that projects $\mathbf{x}^{(\ell)}$ into a new space. This is particularly useful when the samples are not linearly separable in the original $\mathbf{x}^{(\ell)}$ space; ϕ can be used to map the samples into one in which they can be linearly separated. This newly transformed space can be of any arbitrary dimension and yield complex separating hyperplanes that may be computationally expensive to solve. However, by realizing that the decision boundary only requires the dot-product of the vectors in the transformed space, we can replace it with a chosen kernel function and forgo the transformation. This is known as the kernel trick and is what allows SVMs to work with nonlinearly separable data. Section 3.8 describes some possible choices of kernel functions.

Another use of SVMs is that of anomaly detection. This occurs when the training samples are only of a single class, but we need to determine if new samples fall within this class or represent an anomaly. Essentially, the idea is to find a minimal volume hypersphere around the training sample. Any new samples that fall outside the hypersphere are classified as anomalies. This hypersphere is fitted in a transformed space using the kernel trick and is parametrized by its center coordinate $\mathbf{a}$ and its radius R. In a one-class SVM, we cannot maximize the distance between the prior and the unknown non-prior class. Instead, we fit the hyperplane such that all the prior samples are on one side, and we maximize the distance between the hyperplane and the origin of the data space.

$$\min_{R,\mathbf{a}} \|R\|$$

subject to

$$\left\|\mathbf{x}^{(\ell)} - \mathbf{a}\right\|^2 \leq R^2 \quad \forall \ell = 1, \ldots, L \tag{3.136}$$

The decision boundary then becomes

$$f(\mathbf{x}^*) = \mathrm{sgn}\left(\sum_{\ell=1}^{L} \alpha_i K\left(\mathbf{x}^*, \mathbf{x}^{(\ell)}\right) - R^2\right) \tag{3.137}$$

This one-class SVM is useful for detecting when new samples are not part of the class in which the training samples are part of. Chapter 7 has application of one-class SVM for detecting if a prior distribution within a Bayesian context is consistent (or not) with some observed data.

3.7.4. CART: Classification and Regression Trees

3.7.4.1. Single Tree Methods. The basic idea of tree-based methods is simple, yet powerful. Figure 3.21 shows a simple tutorial example with two predictor variables X_1 and X_2. Trees rely on a binary partition of the input space and model Y by means of a constant in the partitioned regions. In other words, the model of Y is piecewise discontinuous. The hierarchy of this partitioning can be represented with a tree-like topology, such as Figure 3.21. The estimate for Y is simply the average of the $y^{(\ell)}$ in each region. The main question is, therefore, how to split the input space in these regions (what is the topology of the tree?).

For regression, the tree model is

$$y = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m) \tag{3.138}$$

which is simply a linear combination of discrete indicator functions (I) over M regions R_m. Based on the observations $(\mathbf{x}^{(\ell)}, y^{(\ell)})$, $\ell = 1, \ldots, L$, a regression model is then estimated by calculating the coefficients as follows:

$$\begin{aligned} \hat{c}_m &= \frac{1}{\#\left(y^{(\ell)} | \mathbf{x}^{(\ell)} \in R_m\right)} \sum_{y^{(\ell)} | \mathbf{x}^{(\ell)} \in R_m} y^{(\ell)} \\ \Rightarrow \quad \hat{y} &= \sum_{m=1}^{M} \hat{c}_m I(\mathbf{x} \in R_m) \end{aligned} \tag{3.139}$$

Basically, this is simply the average of the sample values of y in each region. For categorical variables that are

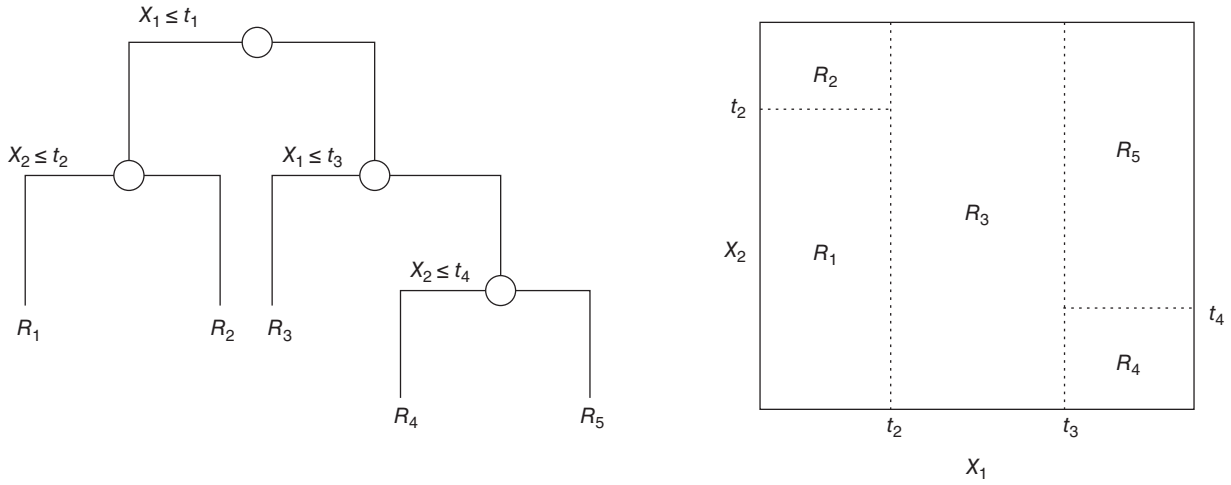


Figure 3.21 Example of a regression tree with two variables and four splits.

ordinal, the splitting occurs at discrete instances. However, for categorical variables that are not ordinal (e.g., a geological scenario), one needs to explore all combinatorials of splitting, which may become large.

To find an optimal partition, one could formulate a least-square criterion between the model and the data, then try out every possible partition. This is computationally infeasible. Instead, a greedy algorithm is used where the least squares is defined based on first the split variable then on the split-point for that choice of variable. The reason for this is that sorting through the universe of variables is easier than the possibilities of split-points. The question now is how large to grow the tree: too large of a tree will lead to overfitting the data. The tree size determines the complexity of the model of Eq. (3.138). The main idea here is to grow a large tree, then "prune" the tree. To achieve this, one formulates a new least-square criterion but now adding a regularization term that includes the number of terminal nodes (see *Breiman et al.* [1984] for details).

For classification the same idea is used except that the least-squares criterion is dropped in favor of a more suitable one for classification. For classification, one models in each region R_m the estimated proportions of each class $y \in \{k = 1, \ldots, K\}$, as $\hat{p}_{mk}$. In such regions, the class with the largest proportion is taken as the estimated class. An example of a suitable criterion (instead of least squares) for establishing an optimal tree is, for example, to minimize entropy

$$E = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}) \tag{3.140}$$

Consider now the application of the regression tree to the hydro case of Section 3.1. The aim here is to emulate the forward model of predicting contaminant arrival time (y) using six input (predictor) variables. Four parameters related to the hydraulic conductivity, K_{mean}, K_{sd}, K_{range}, and K_{Cov}, and two parameters related to boundary conditions, H_{rivGrad}, H_{range}, are used. The tree model is shown in Figure 3.22, split first on K_{mean}, then on H_{rivGrad}, and so on.

3.7.4.2. Boosted Trees. As shown in Table 3.2, the prediction power of trees is lesser than most other methods, they seldom have more accuracy than what can be achieved with the data itself [*Hastie et al.*, 2009]. Figure 3.23 shows that in the hydro case the prediction performance of regression of arrival time in terms of correlation with a test set is only 0.43. This is partly due to the piecewise continuous nature of the model. It is, therefore, considered a weak classifier, meaning it would not do much better than random selection. A powerful method for dealing with such cases is "boosting." The idea of boosting is that it uses outcomes of weak classifiers and produces a powerful "committee" by combining the strengths of each weaker classifier (the whole is better than each of the individuals). The idea of boosting is very simple. It works for both regression and classification, but consider classification as an example here. A classifier, such as a tree, takes some input $\mathbf{x}$ and classifies it using $y(\mathbf{x})$, with possible class outcomes $k = 1, \ldots, K$. The idea of boosting is to generate a sequence of classifiers on repeatedly modified versions of the same data.

Intuitively it works as follows. Consider a data set $(\mathbf{x}^{(\ell)}, y^{(\ell)})$, $\ell = 1, \ldots, L$ of i.i.d. samples, meaning each has equal weight $w_\ell = 1/L$. A base classifier, such as the tree, will perform better on certain pairs $(\mathbf{x}^{(\ell)}, y^{(\ell)})$ than on other pairs. Boosting consist of generating a new tree classifier but now using an error model that has increased weights on

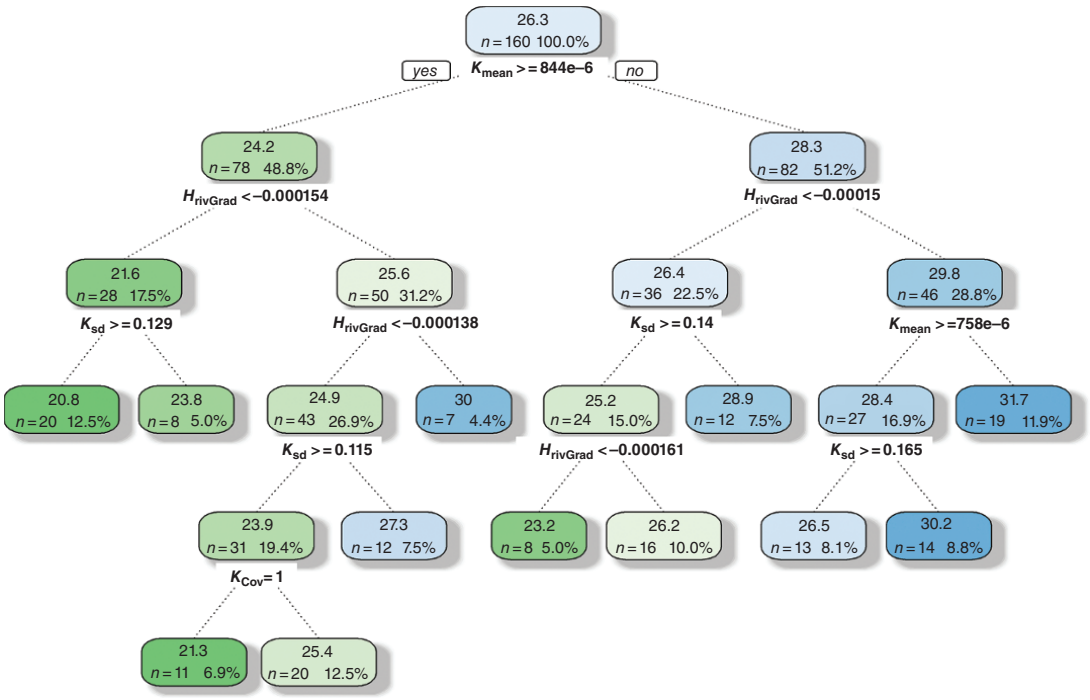


Figure 3.22 Regression tree for the case of Section 3.1, involving six variables and 200 runs of the model. The color is the average arrival time. The color indicates the deviation from the global average (26.3).

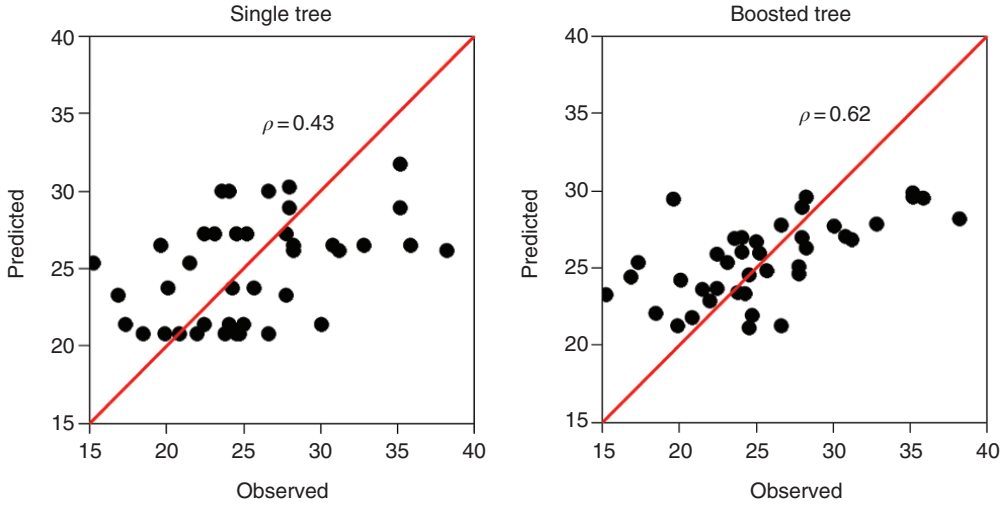


Figure 3.23 Application of a single tree and boosted tree to a test set, shown are observed and predicted arrival time.

samples with large errors. The classifier is, therefore, forced to concentrate on samples that are hard to classify. Each of the generated classifiers in this sequence is still weak classifiers (increasing weights may also decrease performance on others). Therefore, the ultimate classifier is taken as a weighted combination of the sequence of classifiers.

Mathematically, this means that a specification of error is needed for classifier y_m in the sequence $m = 1, \ldots, M$

$$\text{err}(m) = \frac{\sum_{\ell=1}^{L} w_\ell I\left(y^{(\ell)} \neq y_m\left(\mathbf{x}^{(\ell)}\right)\right)}{\sum_{\ell=1}^{L} w_\ell} \tag{3.141}$$

that this error is turned into a weight for that classifier y_m

$$\alpha_m = \log\left(\frac{1-\text{err}(m)}{\text{err}(m)}\right) \tag{3.142}$$

and that this weight and the error are used to determine the re-weighted samples

$$w_\ell \leftarrow w_\ell \exp\left(\alpha_m I\left(y^{(\ell)} \neq y_m\left(\mathbf{x}^{(\ell)}\right)\right)\right), \ell = 1, \ldots, L \tag{3.143}$$

for some initial set of weights $w_\ell = 1/L$. The final classifier is then

$$y(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) \tag{3.144}$$

A boosted tree model is, therefore, simply a weighted sum of trees. Figure 3.23 illustrates the difference in performance between a single tree and a boosted tree for the hydro regression problem. Here 80% of the data (160 runs) are used to establish the tree, while 20% (40 runs) are withheld to evaluate the tree. Even if improved, the rather low correlation also shows that other variables, not used in fitting the tree, affect the response, in particular the heterogeneity of the hydraulic conductivity. In Chapter 4, we will introduce methods that can include these variability tree methods and hence even further their performance.

Tree methods span a large variety of ideas on the same theme. Bagging or bootstrap aggregation with trees consists of generating bootstrap samples of the data set, then averaging the estimator generated with these bootstrap sample. We will discuss *The Bootstrap* in Section 3.13. Unlike bagging, boosting, a so-called committee of weak learners, varies over time. Another modification of boosting and bagging are *random forests*. Here we also generate bootstrap samples of the data, but in addition, when growing the tree, at each terminal node of the tree, we select randomly a subset of variables of the N variables x_n and pick the best split-point amongst that subset. This creates an ensemble of trees (a forest). In a regression problem, one simply averages the trees.

3.7.4.3. Sensitivity. The topic of "sensitivity" of predictor variables on predictants will be extensively treated in Chapter 4. Regression trees constitute one such method since it allows ranking the relative importance (or broadly, sensitivity) of variables $\mathbf{X}$ in predicting Y. *Breiman et al.* [1984] proposed the following quantification of the importance for each variable x_n:

$$I_n^2 = \sum_{j=1}^{J-1} i_j^2 I(v(j) = n) \tag{3.145}$$

To explain this intuitively, consider Figure 3.24. The summing here is over internal nodes, a total of $J-1$. The importance of a variable is quantified by how much the square error improves when splitting the tree on that variable. For example, in the first internal node $j = 1$, we split on the second variable $v(1) = 2$, and calculate how much

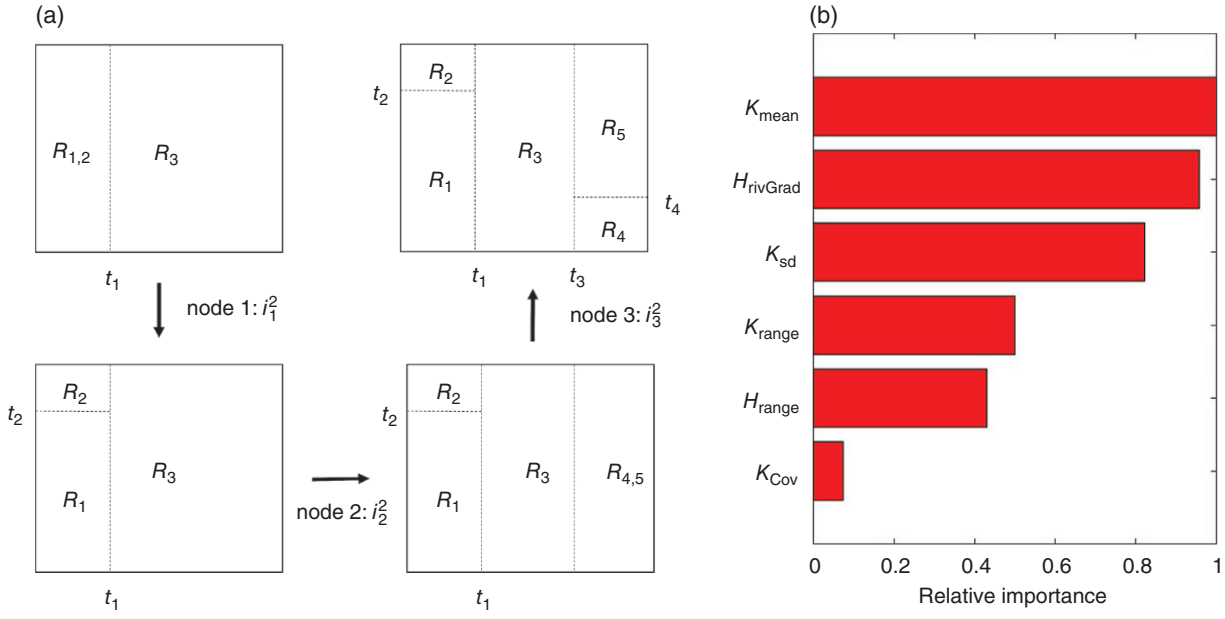


Figure 3.24 (a) Example calculation of variable importance (sensitivity) for the case of Figure 3.21 using regression trees. (b) Relative importance (sensitivity) using boosted tree model for the arrival time response. Relative importance is expressed in percent.

the square error improves as i_2^2. This is continued over all internal nodes, for Figure 3.24, we therefore get

$$I_1^2 = i_2^2 \quad I_2^2 = i_1^2 + i_3^2 \tag{3.146}$$

In Chapter 4, we will use this basic notion and extend it to perform sensitivity analysis on much more complex situations than a few simple scalars and develop methods of sensitivity for objects such as curves, maps, or volumes, given any type of input variable.

When using boosted trees, the relative importance is averaged over all trees in the sequence. Figure 3.24 shows the application of this to the arrival time response case. The mean hydraulic conductivity together with the river gradient have the largest relative importance.

3.7.5. Gaussian Process Regression: Kriging

3.7.5.1. Introduction. Least-square problems comprise a large family of statistical learning and inference problems. The "square" in the names is relevant as differentiation of squared functions lead to linear functions, which then allows for straightforward linear algebra. In addition, the Gaussian distribution contained a squared function (after taking the log), leading to a preponderant role of the Gaussian family of distributions in least-square problems. In presenting these methods, we provide a survey of two literature, which treat the same problem: Gaussian process regression (used in the statistical community) and simple kriging (developed somewhat independently in the geostatistical community).

In uncertainty quantification, least-square problems take an important role, in particular when relationships between models and data or data and prediction are linear. In all such cases, the inference problem is reduced to estimating first- and second-order statistics, in particular conditional means, variance, and covariance of, for example, the model parameters given the data observation. Closed form expressions for the posterior distribution modeling the uncertainty are available in such cases. Evidently, most real-world problems are not Gaussian, nor linear, and hence more advanced methods rely on extending these linear method to more general cases, which we present in Chapter 6.

3.7.5.2. What Is a Gaussian Process?. A Gaussian process is a stochastic process defined over a continuous domain of some finite dimension (e.g., 1D-time, 2D/3D = space). The stochastic nature means that we do not know their outcome at each point of the domain, but we assume it has a Gaussian distribution as a model of uncertainty of that unknown value. We also assume that any finite set of unknown values follows a multivariate Gaussian distribution and, as a consequence, any finite set of linear combinations of unknown values is also multivariate Gaussian. Note that Gaussian processes can be defined over any high-dimensional domain, not just 1D, 2D, or 3D. In 3D, a Gaussian process is also termed a Gaussian "random field." For example, we may have some unknown model variables that are spatially distributed over a grid:

$$\mathbf{m} = \left(m(\mathbf{s}_1), \ldots, m\left(\mathbf{s}_{N_{\text{gr}}}\right)\right) \tag{3.147}$$

with location in space $\mathbf{s}_n$. Consider now any pair of unknown model variables, then the covariance function is

$$C(\mathbf{s}_n, \mathbf{s}_{n'}) = \text{cov}(m(\mathbf{s}_n), m(\mathbf{s}_{n'})) \quad \forall n, n' \tag{3.148}$$

under stationarity assumptions over the domain this reduces

$$C(\mathbf{s}_n, \mathbf{s}_{n'}) = \text{cov}(\mathbf{s}_n - \mathbf{s}_{n'}) \quad \forall n, n' \tag{3.149}$$

with $\mathbf{s}_n - \mathbf{s}_{n'}$ the distance between these two locations in the domain. In geostatistics, one usually uses the variogram instead of the covariance:

$$\gamma(\mathbf{s}_n - \mathbf{s}_{n'}) = \text{var} - \text{cov}(\mathbf{s}_n - \mathbf{s}_{n'}); \quad \text{var} = \text{cov}(\mathbf{0})$$

The Gaussian process is completely defined by the second-order statistics, mean, and covariance function (or variogram). Common covariance functions used are exponential, Matern, or linear, each leading to Gaussian processes with different characteristics. Examples of some realizations of Gaussian processes are shown in Figure 3.25.

An important property of a Gaussian process relates to how Gaussian processes or data that follow such processes can be orthogonalized, meaning having their correlation removed (as was done using PCA). More specifically, consider the orthogonal decomposition of the covariance function as follows (Mercer's theorem):

$$C(\mathbf{s}_n, \mathbf{s}_{n'}) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\mathbf{s}_n) \psi_j(\mathbf{s}_{n'}) \tag{3.150}$$

where $\psi_j(\mathbf{s})$ are termed eigen-functions and λ_j eigenvalues such that

$$\lambda_j \psi_j(\mathbf{s}_n) = \int_{-\infty}^{\infty} C(\mathbf{s}_n, \mathbf{s}_{n'}) \psi_j(\mathbf{s}_{n'}) d\mathbf{s}_{n'} \tag{3.151}$$

This looks similar to PCA (eigen-decomposition of the covariance matrix) but now written with integrals and functions instead of sums and vectors. Mercer's theorem can be used to turn a Gaussian process into a sum of uncorrelated random variables.

$$X(\mathbf{s}_n) = \sum_{j=1}^{\infty} \sqrt{\lambda_j} \psi_j(\mathbf{s}_n) Z_j \quad Z_j \sim N(0,1) \tag{3.152}$$

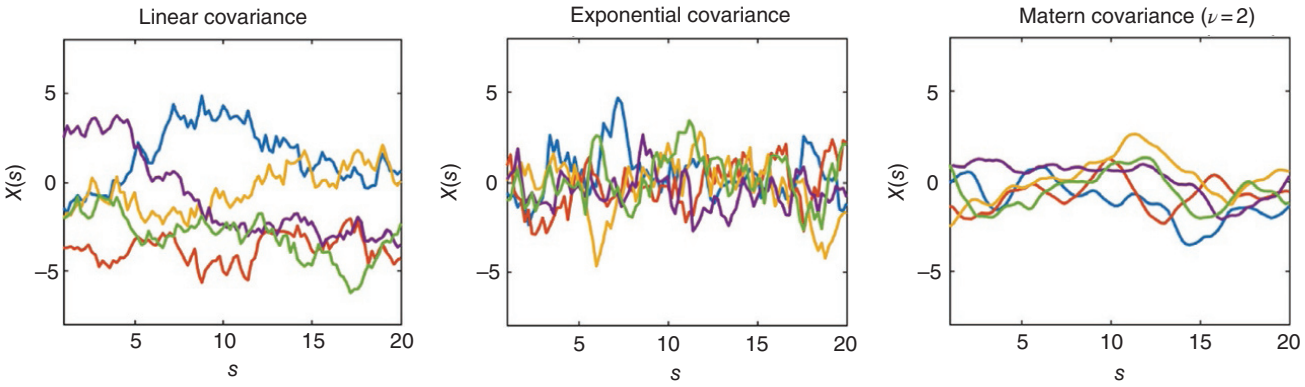


Figure 3.25 Five realizations of Gaussian processes for different covariance models.

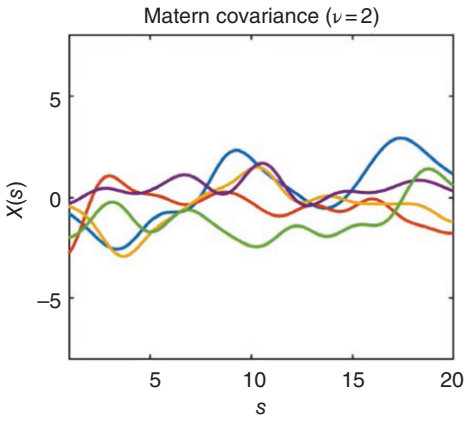


Figure 3.26 Example of approximate realizations of a Gaussian process generated using Karhunen–Loeve expansion (compare with the left figure in Figure 3.25).

Within the context of this book $X(\mathbf{s})$ can represent anything, whether model, data, or prediction variable, hence of any dimension. This decomposition is also termed the Karhunen–Loeve expansion, which is basically a generalization of SVD (PCA) to continuous spaces. A Gaussian process can be approximated by a finite sum by truncating the sum based on decreasing eigenvalues (see Figure 3.26):

$$X(\mathbf{s}_n) \cong \sum_{j=1}^{J} \sqrt{\lambda_j}\psi_j(\mathbf{s}_n)Z_j \quad Z_j \sim N(0,1) \tag{3.153}$$

Generating (approximate) realizations of the Gaussian process can now proceed simply by drawing Gaussian deviate z_j.

3.7.5.3. Prediction with Gaussian Processes. Consider first the general problem of predicting some unknown value, assuming that the unknown values as well as any observed value were somehow generated from a Gaussian process. Note that this cannot be verified [*Mariethoz and Caers*, 2014], since we do not have any replicates of the process. Still, it is interesting to study prediction (and hence uncertainty quantification) under such conditions.

Suppose we have observed the process at certain points $x(\mathbf{s}_1),\ldots,x(\mathbf{s}_N)$ and we want to predict X at some desired location $\mathbf{s}_0$ (with unobserved value of the process). Given the assumption of a Gaussian process, we assume the random vector $(X(\mathbf{s}_1),\ldots,X(\mathbf{s}_N),X(\mathbf{s}_0))$ is multivariate Gaussian. We also assume for convenience that the mean of the process is zero (and known). The covariance matrix of that multivariate Gaussian distribution is partitioned as follows:

$$K_+ = \begin{pmatrix} K & \mathbf{k} \\ \mathbf{k} & k_0 \end{pmatrix} \tag{3.154}$$

with

K : covariance between any two data locations
$\mathbf{k}$: vector of covariance between any data location and the unknown
k_0 : (prior) variance at the location to be predicted

The multivariate distribution of $(X(\mathbf{s}_1),\ldots,X(\mathbf{s}_N), X(\mathbf{s}_0))$ is completely known and needs to be conditioned on specific observations $(x(\mathbf{s}_1),\ldots,x(\mathbf{s}_N))$. The resulting conditional distribution of $X(\mathbf{s}_0)$ is also Gaussian and has conditional mean and covariance [*von Mises*, 1964]

$$\begin{aligned} E\left[X(s_0)|x(\mathbf{s}_1),\ldots,x(\mathbf{s}_N)\right] &= \mathbf{k}^T K^{-1}\mathbf{x} \\ \mathbf{x} &= (x(\mathbf{s}_1),\ldots,x(\mathbf{s}_N)) \\ \mathrm{var}\left(X(\mathbf{s}_0)|x(\mathbf{s}_1),\ldots,x(\mathbf{s}_N)\right) &= k_0 - \mathbf{k}^T K^{-1}\mathbf{k} \end{aligned} \tag{3.155}$$

Important to note is that the expected value is a linear combination of the data and that the conditional variance is not a function of the observed values. Instead, it uses the same linear combination ($\mathbf{k}^T K^{-1}$), but it is now applied to the covariances $\mathbf{k}$ and compared (subtracted) from the

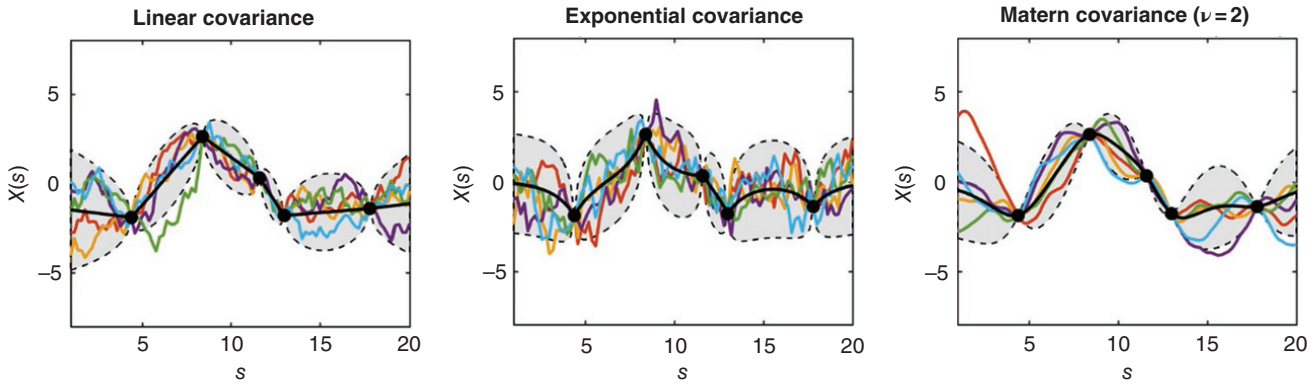


Figure 3.27 Five realizations of Gaussian processes for different covariance models that are conditioned to available data (represented as black dots). The black line represents the mean value and the shaded area represents the 95th confidence interval. This illustrates that Gaussian process regression (kriging) is an exact interpolator.

prior variance (not knowing anything, except that the process is Gaussian with some variance). A useful property of this result is that prediction is exact at the data locations. This means that the expected value, as function of **s**, interpolates exactly the data, as shown in Figure 3.27. This property makes Gaussian process regression an excellent candidate as a surrogate model to emulate a computer model based on a few sample runs. This form of regression will exactly replicate the output of the sample runs, and hence form an interpolator for runs that have not been evaluated, yet its linear formulation avoids overfitting, as compared to nonlinear models (such as ANN).

3.7.5.4. Generalized Linear Regression. In the previous section, we focused on the general problem of predicting with a Gaussian process. It turned out that the conditional expectation of some unknown given observations of the process elsewhere is a linear combination of the observations. The previous method is one of the most general forms of linear prediction in the sense that it uses knowledge about the Gaussian process in terms of a covariance function and uses the covariance between the predictors (the knowns) and the covariance between predictors and the predictant. The resultant weights are, therefore, functions of this covariance.

We now turn to a specific form of linear regression and the particular Gaussian process derived from it (hence we start from specifying the regression first, then make links with Gaussian processes). Consider fitting the data using a generalized linear model:

$$y(\mathbf{s},\mathbf{w}) = \sum_{j=1}^{J} w_j \phi(\mathbf{s}_j) \tag{3.156}$$

We use the classical regression notation of y for predictant. Essentially, we are trying to fit a surface using some observations but within a statistical framework. To do so, we introduce (again) a Gaussian assumption, now not on the unobserved process but on the weights. The weights are considered random variables $\mathbf{W}$ that need to be estimated. Additionally, we consider this regression within a Bayesian context, meaning we assume a prior distribution on the weights, namely a multivariate Gaussian with mean zero and covariance C_w:

$$\mathbf{W} \sim N(\mathbf{0}, C_w) \tag{3.157}$$

Observations may be subject to noise. To model this, we assume the following observation model with random uncorrelated and unbiased error

$$t(\mathbf{s}) = y(\mathbf{s}) + \varepsilon \quad \mathrm{var}(\varepsilon) = \sigma_\varepsilon^2 \tag{3.158}$$

and hence the observations $(t(\mathbf{s}_1), \ldots, t(\mathbf{s}_N))$. We can state a likelihood model for the data under the model of Eq. (3.158), namely

$$L(t(\mathbf{s}_1),\ldots,t(\mathbf{s}_N)|\mathbf{w}) = \frac{1}{\sqrt[N]{(2\pi\sigma_\varepsilon^2)}} \prod_{n=1}^{N} \exp\left(-\frac{(t(\mathbf{s}_n) - y(\mathbf{s}_n,\mathbf{w}))^2}{\sigma_\varepsilon^2}\right) \tag{3.159}$$

According to Bayes' rule, the posterior of the weights is also Gaussian (because both prior and likelihood are Gaussian, with mean

$$E[\boldsymbol{W}|t(\mathbf{s}_1),\ldots,t(\mathbf{s}_N)] = \frac{1}{\sigma_\varepsilon^2}\left(C_w^{-1} + \frac{1}{\sigma_\varepsilon^2}\phi^T\phi\right)\phi^T\mathbf{t} \tag{3.160}$$

with

$$\mathbf{t} = (t(\mathbf{s}_1),\ldots,t(\mathbf{s}_N)) \tag{3.161}$$

and the design matrix

$$\phi = \begin{pmatrix} \phi_1(\mathbf{s}_1) & \cdots & \phi_J(\mathbf{s}_1) \\ \vdots & & \\ \phi_1(\mathbf{s}_N) & \cdots & \phi_J(\mathbf{s}_N) \end{pmatrix} \tag{3.162}$$

This classical solution also has an interpretation in terms of Gaussian processes. The relationship between the weight space view (weights as RV) and the Gaussian process view can be established by choosing C_w to approximate a Gaussian process. To establish that link, we consider unknown and data variables (observations) $(T(\mathbf{s}_1), \ldots, T(\mathbf{s}_N), Y(\mathbf{s}_0))$ to be multivariate Gaussian. Since Y is corrupted by noise, we consider first the noise-free variables $(Y(\mathbf{s}_1), \ldots, Y(\mathbf{s}_N), Y(\mathbf{s}_0))$. Because the Ys are linearly related to the Ws, see Eq. (3.158), a relationship between the covariance of Y and W is as follows:

$$C_y = \phi_0 C_w \phi_0^T \tag{3.163}$$

with the extended design matrix now including the evaluation of the basis functions ϕ at the location to be estimated $\mathbf{s}_0$:

$$\phi_0 = \begin{pmatrix} \phi_1(\mathbf{s}_1) & \cdots & \phi_J(\mathbf{s}_1) \\ \vdots & & \vdots \\ \phi_1(\mathbf{s}_N) & \cdots & \phi_J(\mathbf{s}_N) \\ \phi_1(\mathbf{s}_0) & \cdots & \phi_J(\mathbf{s}_0) \end{pmatrix} \tag{3.164}$$

The extension to corrupted measurements then simply involves adding variance to the diagonal, except for the last row/column since this involves the true unknown (not corrupted with noise evidently):

$$(T(s_1),\ldots,T(s_N),Y(s_0)) \sim N\left(0, \phi_0 C_w \phi_0^T + E\right) \tag{3.165}$$

with

$$E = \begin{pmatrix} \sigma_\varepsilon^2 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \sigma_\varepsilon^2 & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \tag{3.166}$$

Again, because the multivariate normal distribution is now stated, the conditional mean and variance can be expressed as

$$\begin{aligned} E[Y_0|t(\mathbf{s}_1),\ldots,t(\mathbf{s}_N)] &= \phi_0^T C_w \phi^T \left(\phi C_w \phi^T + \sigma_\varepsilon^2 I_N\right)^{-1} \mathbf{t} \\ \mathrm{var}(Y_0|t(\mathbf{s}_1),\ldots,t(\mathbf{s}_N)) &= \phi_0^T C_w \phi_0^T \\ &\quad - \phi_0^T C_w \phi^T \left(\phi C_w \phi^T + \sigma_\varepsilon^2 I_N\right)^{-1} C_w \phi_0 \end{aligned} \tag{3.167}$$

Williams [1999] shows that expressions (3.155) and (3.167) are identical, meaning that the weight-space (W) approach and the function-space approach (Y) are equivalent. The difference lies in computational complexity. In the W-approach one needs to invert a $J \times J$ system (size of the approximation), while in the Y-approach the inversion is of an $N \times N$ matrix (size of data). Recall a similar duality in Section 3.4.4 between sample size space and model size space.

3.7.5.5. History of Applications of Gaussian Process Regression. Several communalities with different views of the same problem exist that invoke the Gaussian process of the unknown phenomenon being estimated (conditional expectations). Historically, these methods have been developed somewhat independently in a number of different areas (statistics, geostatistics, machine learning). Although the basic theory still goes back to Wiener and Kolmogorov, linear prediction is also well known in the geostatistics field [*Matheron*, 1970; *Journel and Huijbregts*, 1978; *Cressie*, 1993] as kriging. Kriging was established as the best linear unbiased estimator:

$$y(\mathbf{s}_0) = \sum_{n=1}^{N} w_n t(\mathbf{s}_n) = \mathbf{w}^T \mathbf{t} \tag{3.168}$$

In geostatistics, noise σ_ε^2 is modeled as the so-called nugget effect. The weights are derived based on a minimum estimation variance criterion, resulting in

$$\mathbf{w} = \mathbf{k}^T K^{-1} \tag{3.169}$$

This solution "identifies" the covariance, meaning that the covariance between the estimate $Y(\mathbf{s}_0)$ and the observations is the same as (identifies) the modeled covariance, based on the observations. The establishment of the kriging equation, initially, did not invoke any Gaussian process. However, simple kriging can be regarded as the conditional expectation of an unknown value of a Gaussian process given observations. Dual kriging [*Goovaerts*, 1997] is another form to express the same kriging, but now the unknown is written as a linear combination of covariance functions:

$$y(\mathbf{s}_0) = \sum_{n=1}^{N} w_n C(\mathbf{s}_0 - \mathbf{s}_n) \tag{3.170}$$

The weights are now established by means of identification with the observed values (the exact interpolator property):

$$\mathbf{w} = \mathbf{t}^T K^{-1} \tag{3.171}$$

Equation (3.156) looks like Eq. (3.170), but it is now written with covariance functions as basis functions. These covariance functions can be estimated from the observations, providing a method for inferring such basis functions. This is typically possible in 3D, but it becomes more difficult in higher dimensions because of emptiness of high-dimensional space and the limited amount of data for such inference. In such cases, inference can be made based on likelihood methods [*Diggle and Ribeiro*, 2007], rather than least-square fitting of covariance or variograms [*Cressie*, 1985].

3.7.5.6. Linear Inverse Problems and Kriging. The solution of linear inverse problems has many applications in the subsurface, either where the relationship between data variables and model variables is linear, or where the linear problem is solved iteratively as part of a larger nonlinear inversion. In this section, we define what a linear inverse problem is and how it is related to Gaussian process regression (kriging) [see *Hansen et al.*, 2006]. A linear inverse problems involves a linear relationship between data variables and model variables

$$\mathbf{d} = G\mathbf{m} \tag{3.172}$$

The aim is to invert (estimate/regress) the model variables from observed data. In this sense, this is an extension of Eq. (3.168) where one model variable is a linear combination of the observed data, but model and data variables are of the same type (although co-kriging can be used to extend the regression to any type, see *Goovaerts* [1997]).

The linear inverse problem can be solved within a Bayesian framework. Just as the case in Gaussian processes, we assume (a priori) that the model parameters follow a multivariate Gaussian distribution with some prior covariance C_m. If the models are defined on a grid with a number of cells N_{grid}, then the observed data (of size N_{data}) is simply

$$\mathbf{d}_{\text{obs}} = \left(d\left(\mathbf{s}_{(1)}\right),\ldots,d\left(\mathbf{s}_{(N_{\text{data}})}\right)\right) = \left(m\left(\mathbf{s}_{(1)}\right),\ldots,m\left(\mathbf{s}_{(N_{\text{data}})}\right)\right) \tag{3.173}$$

with $\mathbf{s}_{(n)}$, $n = 1, \ldots, N_{\text{data}}$ the locations where observations are available. As a result, the operator G simply contains ones and zeros as elements:

$$\begin{aligned} &G_{n'n} = 1 \text{ if } \mathbf{s}_{(n')} = \mathbf{s}_{(n)}, \text{ zero else} \\ &n = 1,\ldots,N_{\text{grid}}; n' = 1,\ldots,N_{\text{data}} \end{aligned} \tag{3.174}$$

G identifies grid locations with data locations. Equation (3.172), however, states that data variables can be forward modeled by any linear combination (not just ones and zeros) of model variables:

$$d_{n'} = \sum_{n=1}^{N_{\text{grid}}} g_{n'n} m(\mathbf{s}_n); \quad n' = 1,\ldots,N_{\text{data}} \tag{3.175}$$

In terms of physical modeling, this applies to a limited set of forward models in the subsurface such as pressure equations (as function of permeability) or tomography problems Arrival times are linear combinations of model velocities (which in itself is an approximation to the full physics in such problems). While traditionally such problems have been solved using least squares, such methods ignore any prior covariance and hence solutions tend to be too smooth (certainly smoother than the actual reality). The prior covariance on the model variables allows injecting any information about the spatial distribution of these properties as quantified by a Gaussian process. The solution to this problem are provided in *Tarantola* [1987, p. 66] and summarized here by listing the expression of the posterior mean $E[\mathbf{M}|\, d_{\text{obs}}]$ of the entire model $\mathbf{m}$ and the posterior covariance $C_{M|d_{\text{obs}}}$:

$$\begin{aligned} E[\mathbf{M}|\, d_{\text{obs}}] &= E[\mathbf{M}] + C_M G^T (G C_M G^T + C_D)(d_{\text{obs}} - G E[\mathbf{M}]) \\ C_{M|d_{\text{obs}}} &= C_M - C_M G^T (G C_M G^T + C_D)^{-1} G C_M \end{aligned} \tag{3.176}$$

Like Gaussian process regression, the posterior covariance is not a function of the observed data. C_D is the covariance of the data variables, which under a simple error model becomes

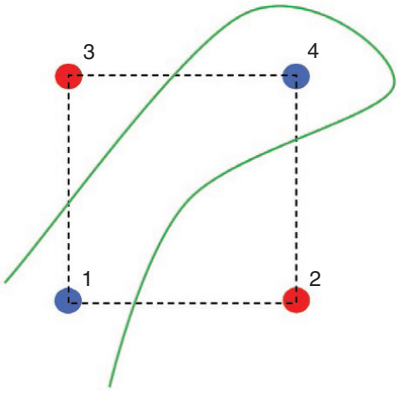
$$C_D = \begin{pmatrix} \sigma_\varepsilon^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_\varepsilon^2 \end{pmatrix} \tag{3.177}$$

Conditioning to linear averages in geostatistics is also known as block kriging [*Journel and Huijbregts*, 1978]. The "blocks" do not need to be square or compact, the term refers to estimating mining blocks (averages) from point data, but the problem can be reversed to estimating points from block data. "Block data" simply refers to some linear averaging. Block kriging requires calculating covariances of the linear averages, covariances between the linear averages and the unknown, which are then plugged into Eq. (3.176). All these covariance can be calculated from C_M. The matrix C_D reflects the so-called nugget effect.

3.8. KERNEL METHODS

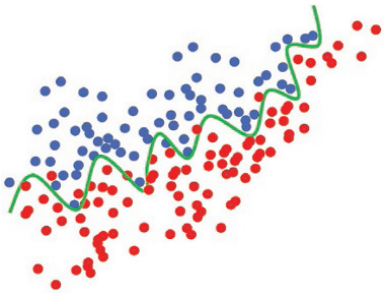
3.8.1. Introduction

Consider the following simple problem (see Figure 3.28). The aim is to cluster the two red points in one group and the two blue points in another group. To achieve this would require a complex discriminant function. A discriminant function is a function that divides space into two parts, to "discriminate" one part from the other. Obviously, this is useful for classification where any point in one half are then considered to belong to group "red" and the other to group "blue." Using highly nonlinear discriminant function can be problematic because (i) finding expressions for such function may not be trivial and (ii) overfitting may occur quite rapidly. This is shown in Figure 3.28. The discriminant function fits the problem of discriminating the blue points from the red points almost perfectly; however, that discrimination will have poor performance when applied to yet



Distance d

	1	2	3	4
1	0	1	1	$\sqrt{2}$
2	1	0	$\sqrt{2}$	1
3	1	$\sqrt{2}$	0	1
4	$\sqrt{2}$	1	1	0



$k = 1 - \exp(-d)$

	1	2	3	4
1	0	0.86	0.86	0.94
2	0.86	0	0.94	0.86
3	0.86	0.94	0	0.86
4	0.94	0.86	0.86	0

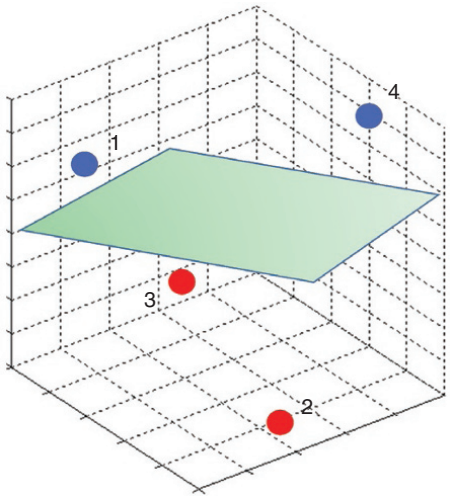


Figure 3.28 A simple classification problem with a nontrivial solution. Linear classification is not possible in this XOR problem (exclusive or). Linear discriminant functions are desirable, in particular when going to higher dimension. The "trick" is to transform the distances to obtain a higher-dimensional space.

unseen blue/red points. This problem is obvious here, but it becomes difficult to diagnose in higher dimensions. An obvious solution is to split the data into a training set and a validation set and use such validation set to avoid overfitting. However, this would not mitigate the problem of what function to choose in the first place, and it may not be applicable when only few samples are available.

For these reasons, linear methods have their appeal, but clearly there is no linear solution to the problem of Figure 3.28, which uses 2D Cartesian space with Euclidean distances. How can we make a linear method work? Take now the example in Figure 3.29 with a complex curve in 2D. Consider now increasing the dimension by one and embedding that curve into a plane (a linear feature!) such that the projection back into 2D still results in the same curve. Figure 3.29 achieves this. One can imagine that if the curve is a helix, then further increasing the dimensions allows embedding the helix into a hyperplane (unfortunately, we cannot show that in a figure!)

The idea, therefore, is to change space using some transformation. However, we will not be transforming coordinates of a Cartesian axis system, as this again would call for complex multivariate transformation functions (and hence we are back to the same problem). To change space, we will change distances to create a new space. Recall that distances can be expressed as dot-products and vice versa (see Section 3.5.2). Now let us go back to our problem in Figure 3.28. Let us first calculate the distance between the four points, listed in the distance table; clearly, the two red points are far from each other than the red from the blue point and vice versa. Now we transform that distance table into a new distance table using the following simple equation:

$$k_{ij} = 1 - \exp\left(d_{ij}\right) \quad i, j = 1, \ldots, 4 \tag{3.178}$$

The exponential function of minus the distance makes objects that are far apart appear closer and objects that are closer further apart. That is exactly what we like to achieve, because it would move the two red points closer, the two blue point closer, and the red points further apart from the blue points. The problem is that this cannot be done in 2D. Why not? This can be explained by calculating the eigenvalues of the distance matrix d and of the matrix k, we find

$$\begin{aligned} d &: \lambda_1 = \lambda_2 = 1; \lambda_3 = \lambda_4 = 0 \\ k &: \lambda_1 = \lambda_2 = 0.44; \lambda_3 = 0.31; \lambda_4 = 0 \end{aligned} \tag{3.179}$$

The distance table has only two positive eigenvalues. This means that we can only create a 1D or 2D projection from the distance table to Cartesian coordinates. The k-table, however, has three positive eigenvalues. This means that we can construct an orthogonal 3D Cartesian space, with its own (3D) Euclidean distance (despite the fact that we

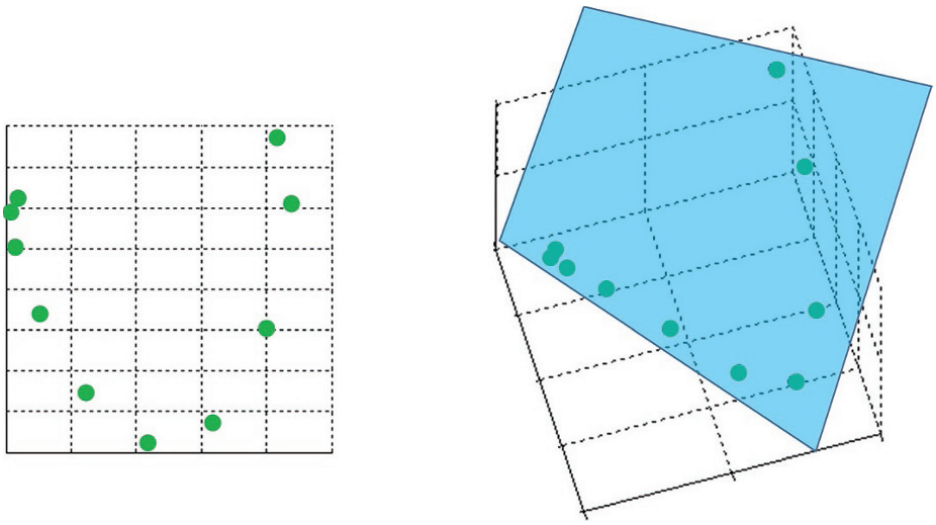


Figure 3.29 Embedding a nonlinear 2D function in a 3D plane, such that projection retrieves the nonlinear function.

only specified 2D distances). In fact, if we perform an MDS projection into this 3D space, we notice how the blue points moved up and the red down. A simple plane (a linear function!) now solves the classification problem, exactly what we set out to achieve.

The problem in Figure 3.28 involves a classification problem, but classification and regression problems are basically variations of the same theme as we discussed in Section 3.7. In classification problems, we seek a discriminant function. In regression, we seek also to estimate a function between predictors and predictants. In the both cases, we have to estimate a function. We saw in Section 3.7 that linear methods, such as Gaussian process regression, can be powerful, if the linear models are adequate to describe the actual complexity of data on which they are applied. In Figure 3.28, we observed how embedding the "data" into a new space, where linear operations apply more readily, is an interesting concept worthwhile pursuing. To understand a bit better how this works for regression problems, we consider two views of the same regression. Consider a data set

$$\left(\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(L)}, y^{(L)}\right)\right) \tag{3.180}$$

with $\mathbf{x}$ a vector of any dimension N of predictors $\mathbf{x} = (x_1, \ldots, x_N)$ (could include spatial location, so we exchange $\mathbf{x}$ and $\mathbf{s}$ in denoting predictors). A linear model is assumed

$$y(\mathbf{x}, \mathbf{w}) = \sum_{n=1}^{N} w_n x_n = \mathbf{w}^T \mathbf{x} \tag{3.181}$$

If we denote, as before,

$$\begin{aligned} X &= \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(L)}\right) \\ \mathbf{y} &= \left(y^{(1)}, \ldots, y^{(L)}\right) \end{aligned} \tag{3.182}$$

Then the classical and primal solution for the least-square weights is (see Eq. (3.130))

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T \mathbf{y} \tag{3.183}$$

An alternative form of presenting the same solution in a dual form is

$$\mathbf{w} = X^T \tilde{\mathbf{w}} \text{ with } \tilde{\mathbf{w}} = \left(X X^T\right)^{-1} \mathbf{y} \tag{3.184}$$

The primal form on $X^T X$ and the dual form relies on $X X^T$. If this looks familiar, then refer to presentation of the data matrices in two ways: the space formed by the dimension of the vector $\mathbf{x}$, using covariances ($X^T X$), or the space formed by sample size with dot-products ($X X^T$). We discussed the advantage of working with $X X^T$ over $X^T X$ in the type of UQ problems we are dealing with. As a result, the presentation of the material that comes next, kernel mapping, can be highly effective in addressing certain UQ problems, in particular those that involve complex priors and nonlinear operations. Basically, we will see that it is easy to extend any linear statistical operation simply by changing dot-products (see Figure 3.30). This will invoke a transformation $\boldsymbol{\varphi}$ of the original Cartesian space; however, an explicit representation of $\boldsymbol{\varphi}$ (a possible high-dimensional function, which as we know, we would like to avoid) will not be required, only a dot-product (just a scalar!). The change in dot-product aims at embedding the data in a space in which linear operations apply more readily (see Figure 3.30).

3.8.2. Kernel-Based Mapping

Consider a general nonlinear mapping between two spaces

$$\boldsymbol{\varphi} : \mathbf{x} \in \mathbb{R}^N \mapsto \mathbf{x} \in \mathbb{R}^M \tag{3.185}$$

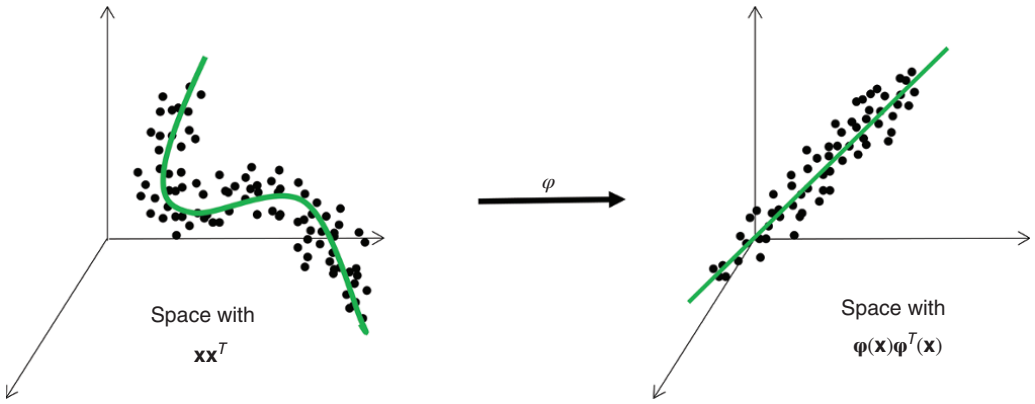


Figure 3.30 Changing dot-product to render linear modeling more appropriate. The space on the right is also termed the "feature space" in machine learning, features being quantified by the kernel function. Features in statistics are termed simply "predictors."

Both N and M could be very large (possibly even infinite) and hence a function $\boldsymbol{\varphi}$ that creates a space where linear methods more readily apply would be difficult to find. Instead, many such linear methods only require knowing a dot-product. In this new space, the dot-product between any two samples $\mathbf{x}^{(\ell)}$and $\mathbf{x}^{(\ell')}$ is

$$g_{\ell\ell'} = \boldsymbol{\varphi}^T\left(\mathbf{x}^{(\ell)}\right)\boldsymbol{\varphi}\left(\mathbf{x}^{(\ell')}\right) \tag{3.186}$$

where the matrix G consisting of elements $g_{\ell\ell'}$ is termed the Gram matrix (or kernel matrix). Then, when considering a new predictor $\mathbf{x}$, we write

$$k_\ell = \boldsymbol{\varphi}^T\left(\mathbf{x}^{(\ell)}\right)\boldsymbol{\varphi}(\mathbf{x}) \tag{3.187}$$

which leads to the definition of a kernel function:

$$k(\mathbf{x},\mathbf{x}') = \boldsymbol{\varphi}^T(\mathbf{x})\boldsymbol{\varphi}(\mathbf{x}') \tag{3.188}$$

Since a dot-product can be related to a distance, in that sense, the kernel matrix contains all the information needed to compute distances, and hence define a metric space. Such space has no orientation; therefore, the kernel matrix is rotationally invariant. Transformations such as rotation are, however, not really important when doing regression or performing classification. Therefore, the kernel matrix is viewed as the information bottleneck, filtering the necessary information to perform "learning" whether this is regression, classification, or orthogonal component analysis. Hence, the "a priori" insight that renders the problem more linear is provided by means of distances. The choice of the kernel function is, therefore, relevant to solve complex problems.

A kernel that will be used throughout this book is the radial basis function (RBF). The RBF kernel function is expressed as

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right) \text{ with } \sigma > 0. \tag{3.189}$$

The performance of the RBF kernel depends highly on its bandwidth σ. If σ is small, the kernel matrix is close to the identity matrix ($K = I$); hence, all the $\mathbf{x}$ will tend to be very dissimilar. On the other hand, large values of σ makes the kernel matrix close to a constant matrix ($K = \mathbf{1}\mathbf{1}^T$), leading to all the $\mathbf{x}$ being very similar. Cross-validation techniques are very popular to estimate the kernel bandwidth in the case of supervised learning. However, for unsupervised learning (such as KPCA), the choice of the bandwidth remains an open question. We will rely on the rule of thumb of *Kwok and Tsang* [2004].

Figure 3.31 provides an example of what a kernel function does on mapping concentrations in feature space (kernel space), for the simple hydro case. The left plot is the MDS plot based on the Euclidean distance (or PCA with covariance) between the original concentration curves; one notices how for small distances the data is strongly clustered but spreads out for larger distances. This is common in classical metric spaces, a few large distances are easier to approximate in lower dimensions than a lot of small distances. The kernel transformation makes these distances more uniform (by increasing dimension). Figure 3.31 shows a nice arrangement of models along a curve-linear feature. Modeling in feature space is preferable over modeling in the original Cartesian space in these types of cases.

3.8.3. Kernel PCA

3.8.3.1. Method. Recall from Section 3.5.1 that PCA performs a projection onto orthogonal bases, derived from the covariance matrix (space of X^TX). This projection onto the n-th eigenvectors can be written as

$$\mathbf{u}_n^T\mathbf{x} = \sum_{\ell=1}^{L} \frac{v_n^{(\ell)}\left(\mathbf{x}^{(\ell)}\right)^T\mathbf{x}}{\sqrt{\lambda_n}} \quad n \le \text{rank}(X) \tag{3.190}$$

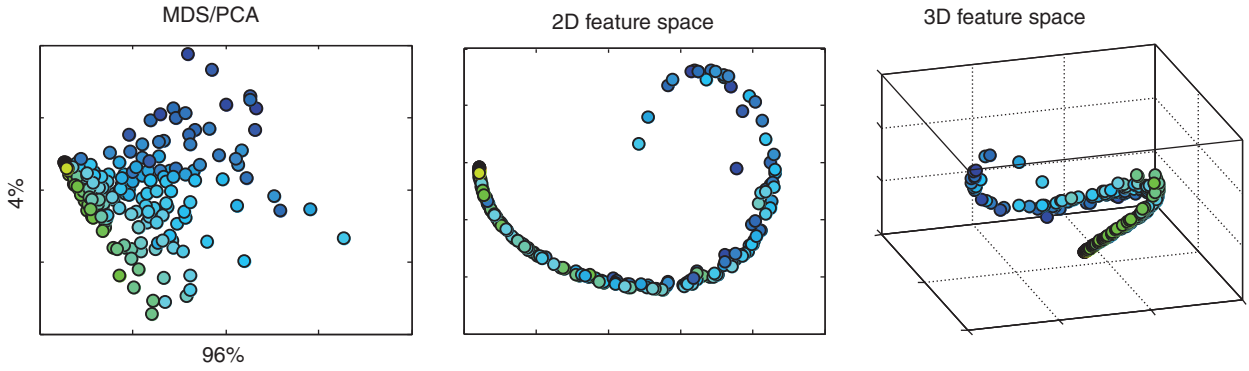


Figure 3.31 Comparison of classical MDS with MDS after kernel transformation. Notice, like in Figure 3.28, the impact of an increase in dimension.

Here we use the duality that exists in using X^TX or XX^T to calculate eigenvalues and eigenvectors $\mathbf{u}$ and $\mathbf{v}$, respectively. We now apply the same kernel trick to turn PCA into kernel PCA (KPCA) by changing the dot-product $\mathbf{x}^T\mathbf{x}$ to $\boldsymbol{\varphi}(\mathbf{x})^T\boldsymbol{\varphi}(\mathbf{x})$ to obtain a projection (now in XX^T space) as

$$\sum_{\ell=1}^{L}\frac{v_n^{(\ell)}\boldsymbol{\varphi}^T\left(\mathbf{x}^{(\ell)}\right)\boldsymbol{\varphi}(\mathbf{x})}{\sqrt{\lambda_n}}=\sum_{\ell=1}^{L}\frac{v_n^{(\ell)}}{\sqrt{\lambda_n}}k\left(\mathbf{x}^{(\ell)},\mathbf{x}\right) \tag{3.191}$$

This leads to the following calculation for KPCA (compare with PCA in Section 3.5.1.2)

$$\begin{aligned}&\text{Calculate } g_{\ell\ell'}=k\left(\mathbf{x}^{(\ell)},\mathbf{x}^{(\ell')}\right),\quad \ell,\ell'=1,\ldots,L\\&\text{Center } g_{\ell\ell'}\leftarrow g_{\ell\ell'}-\frac{1}{L}g_{\ell\bullet}-\frac{1}{L}g_{\bullet\ell'}+\frac{1}{L^2}g_{\bullet\bullet}\\&\text{Decompose } G=V^T\Lambda V\\&\text{Project } \mathbf{y}=\sum_{\ell=1}^{L}\frac{v_n^{(\ell)}}{\sqrt{\lambda_n}}k\left(\mathbf{x}^{(\ell)},\mathbf{x}\right)\end{aligned} \tag{3.192}$$

Note how the centering in the covariance is now replaced by the centering of the dot-product (see Section 3.5.2).

3.8.3.2. The Pre-image Problem. PCA is a bijective projection, meaning that one can uniquely recover the original vector, after projection and reconstruction. This is not the case for KPCA. This makes sense, intuitively: KPCA relies on distances (similarities) between vectors, and not on their exact location in a Cartesian setting. Therefore, reconstructing a new vector based on the original data is not unique. This is the case for UQ in the subsurface where $L \ll N$. For example, imagine knowing the distance between three vectors in 100-dimensional space and we are given a fourth vector for which only the distance to the previous three is specified. Reconstructing this fourth vector is a nonunique problem, many solutions exist. In computer science literature, this is termed a "pre-image problem" [*Schölkopf and Smola*, 2002]. Pre-image means seeking the "image" of an object, given the distances with other objects whose "images" are known.

The difficulty in the pre-image problem is that the mapping function $\boldsymbol{\varphi}$ into the feature space is unknown, nonlinear and nonunique, thus only approximate solutions can be generated. Consider a feature space expansion $\boldsymbol{\Psi}=\sum_{\ell=1}^{L}\alpha^{(\ell)}\boldsymbol{\varphi}\left(\mathbf{x}^{(\ell)}\right)$ and denote $\mathbf{x}^*$ as its approximate pre-image. The pre-image problem attempts to minimize the squared distance in feature space:

$$\min_{\mathbf{x}^*}\|\boldsymbol{\Psi}-\boldsymbol{\varphi}(\mathbf{x}^*)\|^2=\min_{\mathbf{x}^*}\left\|\sum_{\ell=1}^{L}\alpha^{(\ell)}\boldsymbol{\varphi}\left(\mathbf{x}^{(\ell)}\right)-\boldsymbol{\varphi}(\mathbf{x}^*)\right\|^2 \tag{3.193}$$

From the kernel trick, the minimization of Eq. (3.193) is equivalent to minimizing

$$\begin{aligned}\min_{\mathbf{x}^*}\,&k(\mathbf{x}^*,\mathbf{x}^*)-2\sum_{\ell=1}^{L}\alpha^{(\ell)}k\left(\mathbf{x}^*,\mathbf{x}^{(\ell)}\right)\\&+\sum_{\ell'=1}^{L}\sum_{\ell=1}^{L}\alpha^{(\ell)}\alpha^{(\ell')}k\left(\mathbf{x}^{(\ell)},\mathbf{x}^{(\ell')}\right)\end{aligned} \tag{3.194}$$

which formulates a nonlinear optimization problem that is only function of the kernel function and not of $\boldsymbol{\varphi}$. Gradient procedures can be used to minimize this expression. In the particular case of the RBF kernel, a fixed-point iterative approach can be used to find approximate pre-images [*Schölkopf and Smola*, 2002]. The following solution is then obtained:

$$\mathbf{x}_{t+1}^*=\frac{\sum_{\ell=1}^{L}\alpha^{(\ell)}\exp\left(-\left\|\mathbf{x}^{(\ell)}-\mathbf{x}_t^*\right\|^2/2\sigma^2\right)\mathbf{x}^{(\ell)}}{\sum_{l=1}^{L}\exp\left(-\left\|\mathbf{x}^{(\ell)}-\mathbf{x}_t^*\right\|^2/2\sigma^2\right)} \tag{3.195}$$

Unfortunately, the fixed-point iterative method suffers from encountering local minima and tends to be unstable. An interesting property of the fixed-point iterative method is that the resulting pre-image lies in the span of the available data, since the pre-image is simply a weighted sum of $\mathbf{x}^{(\ell)}$.

3.9. CLUSTER ANALYSIS

Cluster analysis is widely used to find hidden structures that may exist in data sets. The aim of clustering is to partition a set of L data points $\mathbf{x}^{(\ell)}$, $\ell = 1, \ldots, L$ into K groups or clusters, based on the similarity between data points: data within a cluster should be similar, and dissimilar to data in other clusters. An ideal cluster is a set of points that is compact and isolated [*Jain*, 2010]. Clustering is used in a variety of applications, such as data mining, image segmentation, and data compression. In subsurface modeling, clustering is sometimes applied to group subsurface models that are similar, given some defined measure of similarity, or for dimension reduction purposes.

3.9.1. *k*-Means

Among many available clustering algorithms, k-means clustering is probably one of the most widely used algorithms because of its simplicity and efficiency. The objective of k-means is to find the cluster configuration that minimizes the squared error over all K clusters:

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}^{(\ell)} \in c_k} \left\| \mathbf{x}^{(\ell)} - \boldsymbol{\mu}^{(k)} \right\|^2 \quad \boldsymbol{\mu}^{(k)} = \frac{\sum_{\mathbf{x}^{(\ell)} \in c_k} \mathbf{x}^{(\ell)}}{|S_k|} \tag{3.196}$$

with $\boldsymbol{\mu}^{(k)}$ the centroids of cluster c_k defined as the mean point of the cluster. $|S_k|$ the number of samples in the cluster c_k. The main steps of the k-means procedure are illustrated in Figure 3.32 and are as follows:

Step 1: Select randomly K centroids $\boldsymbol{\mu}^{(k)}$. These points need not correspond to any of the data points.

Step 2: Find the closest centroid $\boldsymbol{\mu}^{(k)}$ to the point $\mathbf{x}^{(\ell)}$ and assign point $\mathbf{x}^{(\ell)}$ to cluster c_k.

Step 3: Update centroids $\boldsymbol{\mu}^{(k)}$ using the equation above.

Step 4: Repeat Steps 2 and 3 until convergence, that is cluster configuration is stabilized (the squared error is minimized).

In the k-means procedure, the number of clusters K needs to be specified prior to clustering and remain fixed. Methods have been developed to determine the number of clusters (see Section 3.9.4). The k-means algorithm finds a local minima of Eq. (3.196); hence, the clustering results may differ with the choice of different initial clusters centers. This can be mitigated by running k-means multiple times with different initial randomly chosen centroids and selecting the partition with the smallest squared error.

3.9.2. *k*-Medoids

Instead of taking the mean values of the data within a cluster as centroids, k-medoids assigns the cluster center to the most central point of that cluster (referred to as the medoids, the point "in the middle"). Partitioning in k-medoids is still based on the minimization of the sum of the dissimilarities of the data and the cluster centers, but k-medoids does not require the use of a squared Euclidean distance in the calculation of the cost function (Eq. (3.196)). k-medoids algorithms may employ directly any dissimilarity distance matrix. k-medoids clustering is more robust to noise and outliers than k-means.

Among many algorithms for k-medoids clustering, partitioning around medoids (PAM) proposed by *Kaufman and Rousseeuw* [1990] is the most popular. PAM consists of first selecting randomly k-medoids and assigning each data point to the cluster with the closest medoid in terms of the dissimilarity measure. Then, iterating over the data points, each medoid and non-medoid are swapped. If the total cost of the configuration is decreased then the swap is preserved. The algorithm stops when no further permutations improve the quality of the clustering. The major

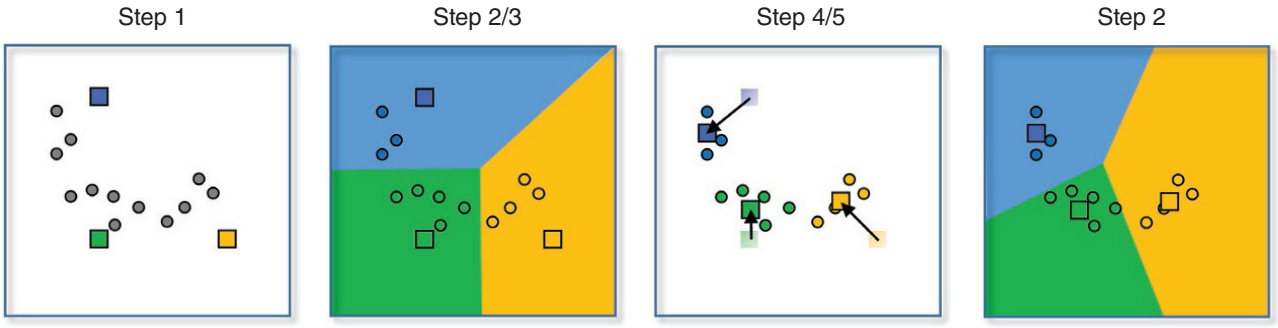


Figure 3.32 An example of partitioning. Image from *Caers* [2011].

drawback of PAM is its high computational cost, which makes k-medoids more costly than k-means. PAM becomes impractical in terms of CPU when the number of data points L or the number of clusters K is large. Its complexity is of the order $O^{K(L-K)^2}$.

3.9.3. Kernel Methods for Clustering

Data may contain nonlinear structures that cannot be easily separated by linear models such as k-means or k-medoids. An example of such a case was given in Figure 3.28. Linear methods such as k-means would perform poorly. One way to tackle this problem is to use kernel clustering (kernel k-means or kernel k-medoids). The use of kernels can, by increasing the dimension of the problem, result in increased linearity and separability. In kernel-based clustering, the data points are first mapped into a high-dimensional space (the feature space), and then clustering is applied in this space. For both kernel k-means and kernel k-medoids methods, the distance in feature space between a point and its centroid/medoid can be computed using only the kernel function. In the case of k-means, a pre-image problem must be solved to obtain the centroid coordinates in the original space.

3.9.4. Choosing the Number of Clusters

3.9.4.1. Silhouette Index. *Kaufman and Rousseeuw* [1990] proposed a "silhouette index" to determine the quality of the clustering. This index can also be used to find the optimal number of clusters K to use in the clustering algorithm. Let $a(\ell)$ be the average distance of a point $\mathbf{x}^{(\ell)}$ to all other points in the same cluster. It measures how well the point $\mathbf{x}^{(\ell)}$ is assigned to its cluster (the smaller, the better). Let $b(\ell)$ represent the minimum of the average distance between $\mathbf{x}^{(\ell)}$ and the points in different clusters:

$$b(\ell) = \min_k d\left(\mathbf{x}^{(\ell)}, c_k\right) \tag{3.197}$$

with $d(\mathbf{x}^{(\ell)}, c_k)$ the average distance between $\mathbf{x}^{(\ell)}$ and all points in c_k ($\mathbf{x}^{(\ell)}$does not belong to c_k). The silhouette index $s(\ell)$ is then defined as follows:

$$s(\ell) = \frac{b(\ell) - a(\ell)}{\max(a(\ell), b(\ell))} \tag{3.198}$$

If $s(\ell)$ is close to one, the data point $\mathbf{x}^{(\ell)}$ is well classified, whereas if $s(\ell)$ is close to zero, it is unclear whether $\mathbf{x}^{(\ell)}$ should belong to its assigned cluster or its neighboring cluster. A negative value is an indication that the data point $\mathbf{x}^{(\ell)}$ has been misclassified. The average value over all data points $\mathbf{x}^{(\ell)}$ is called the average silhouette index and can be evaluated for different number of clusters. The best clustering configuration is achieved when the average silhouette index is maximal. The silhouette index can be plot as a function of the number of clusters. This plot often has an "elbow" shape (and is sometimes referred to as elbow plot). The optimal number of clusters is obtained when the average silhouette index value bends "at the elbow," see for example Figure 3.33.

3.9.4.2. Davies–Bouldin Index. An alternative index to quantify cluster quality and optimal number of clusters is the Davies–Bouldin index [*Davies and Bouldin*, 1979]. It is defined as a function of the ratio of the within cluster scatter to the between cluster separation:

$$DB = \frac{1}{K}\sum_{i=1}^{K} \max_{j \neq i}\left\{\frac{S_i + S_j}{M_{ij}}\right\} \tag{3.199}$$

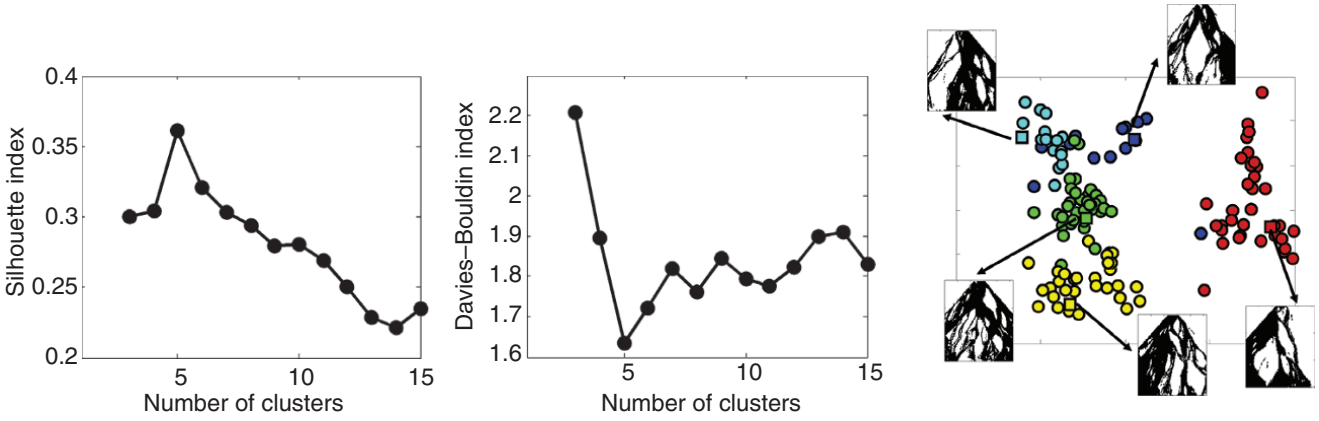


Figure 3.33 Choice of the optimal number of clusters using the silhouette index and Davies–Bouldin index. Five clusters are created using k-medoid, applied on the set of 136 overhead snapshot of a flume experiment. Note that the clustering is applied in high dimension (and not 2D).

where the intra-cluster (within) distance is defined as

$$S_i = \left(\frac{1}{N_{c_i}} \sum_{\ell=1}^{N_{c_i}} \left| \mathbf{x}^{(\ell)} - \boldsymbol{\mu}^{(i)} \right|^p \right)^{\frac{1}{p}} \tag{3.200}$$

with N_{c_i} the number of points assigned to cluster c_i and the cluster separation (inter-cluster distance) between cluster c_i and c_j as

$$M_{ij} = \left(\sum_{n=1}^{N} \left| \boldsymbol{\mu}_n^{(i)} - \boldsymbol{\mu}_n^{(j)} \right|^p \right)^{1/p} \tag{3.201}$$

Usually, $p = 2$. As opposed to the silhouette index, the optimum number of clusters is obtained by minimizing the index with respect to the number of clusters (Figure 3.33). The Davies–Bouldin index is described here for the k-means algorithm, but it could be estimated similarly for k-medoids, by replacing the centroids by the medoids.

3.9.5. Application

Clustering is applied to the set of 136 overhead snapshots of a flume experiment, based on the modified Hausdorff distance between snapshots. This data set will be further discussed in Chapter 5. Both the silhouette index and the Davies–Bouldin index suggest that the optimal number of clusters for this data set is 5. The k-medoid procedure was applied to identify five medoids, which correspond each to one of the snapshots. The five selected images can be considered as representative of the set of 136 images, as defined by the modified Hausdorff distance. This idea is very useful when dealing with a large number of spatial model realization in UQ. Using clustering, one can reduce this large set to a representative smaller set that has the same impact in terms of UQ [see *Scheidt and Caers*, 2009; *Scheidt and Caers*, 2013].

3.10. MONTE CARLO AND QUASI MONTE CARLO

3.10.1. Introduction

In this important section in the context of uncertainty quantification, we discuss various methods of Monte Carlo (MC) simulation. MC is used in many fields of science and engineering and literature is extensive. Here we focus on those methods that are relevant to UQ in subsurface systems. The development of MC goes back to John von Neumann who introduced random numbers generated by a computer to solve problems encountered in the development of atom bomb [*Von Neumann*, 1951]. MC refers to the famous casino in Monaco, where randomness is also used in a perhaps more joyful way than von Neumann's original application.

Broadly, MC uses random sampling to study properties of systems with components that behave in a random fashion [*Lemieux*, 2009]. The idea is simply to simulate on a computer the behavior of a system (in terms of outputs for example) by randomly generating the variables that control/model/describe the system (e.g., the inputs). Then, using the obtained results one can study the system, such as to perform a sensitivity analysis or apply any other statistical analysis or inference. This requires (i) the description of the "model," including a computer code implementing "the model," (ii) specification of distributions on the variables describing the system, (iii) generating samples from these distributions and evaluating the samples using the computer written code, and (iv) statistical analysis and learning from the obtained results. From a purely practical point of view, one needs to define (i) a mathematical model, (ii) a computer implementation of it, and (iii) a way of random sampling and propagating that sampling through the computer code; then observing and analyzing the output of that code, for whatever purpose is deemed relevant. This very broad description encapsulated many specific applications such as the following:

1. *Sampling from a known distribution:* A probability distribution is a mathematical model that can be implemented with a numerical model in a computer program. The inputs are pseudo-random numbers that are passed through this program to generate samples which can then be studied statistically. Underlying all MC methods, therefore, is the generation of these pseudo-random numbers. The pseudo-random number generator itself is a computer code that takes a deterministic input (the seed) and generates an (approximate) sequence of random numbers in the interval [0,1].

2. *Stochastic simulation:* This is generically described as the study of systems that contain stochastic components, broadly represented as $\mathbf{Y} = g(\mathbf{X})$. $\mathbf{X}$ denotes a set of random variables (the stochastic components) that needs to be simulated to generate outputs $\mathbf{Y}$. g is a "transfer function," or "forward model," or "system response model"; whatever flavor is used, it again represents some computer code (hence deterministic). Such computer code can be a simple function or be as complex as a numerical implementation of a partial differential equation that models a dynamic system (e.g., wave equations or flow and transport in porous media).

3. *MC integration:* A specific problem in MC simulation where the goal is to use random sampling to solve a deterministic problem (or problem that has no inherent stochastic component) of the kind:

$$\int_V f(\mathbf{x}) d\mathbf{x} \tag{3.202}$$

MC integration solves this problem by generating random samples of $\mathbf{x}$. For example, if $f(x) = x\,g(x)$ with $g(x)$ a positive function whose integral is unity, then MC integration boils down to estimating the mean of the random variable X.

3.10.2. Sampling from Known Distributions

3.10.2.1. Inversion. The method of inversion goes back to von Neumann. Inversion applies to the cdf, simply as follows, for a continuous distribution:

$$\begin{aligned} u &\leftarrow \text{rand}(\bullet) \\ x &= F^{-1}(u) \end{aligned} \tag{3.203}$$

Obviously, this method depends on the ability to calculate the inverse of the cumulative distribution F. For discrete distributions this becomes

$$\begin{aligned} u &\leftarrow \text{rand}(\bullet) \\ x &= \inf\{y : F(y) \geq u\} \end{aligned} \tag{3.204}$$

If an explicit expression for inf is not available, then this becomes a search problem.

3.10.2.2. Acceptance–Rejection. Sampling through inversion is usually only possible in univariate cases and with fully known cdfs of pdfs. In most cases, it will be impossible to directly sample from a distribution, in particular when we have only the pdf and then usually only up to some normalization constant (that we often cannot compute analytically). Since we cannot directly sample from the target pdf $f(x)$, we will first sample from another pdf $t(x)/T$ (i) that we know how to sample from and (ii) $t(x)$ majors $f(x)$ over the domain of interest, namely there exists some M such that $t(x)M \geq f(x)$. $t(x)$ itself is not a density, but

$$\int t(x)dx = T \tag{3.205}$$

The rejection–acceptance method works as follows:

$$\begin{aligned} &\text{Sample } y \text{ from } t(x)/M \\ &u \leftarrow \text{rand}(\bullet) \\ &\text{If } u \leq \frac{f(x)}{t(x)} \Rightarrow x = y \text{ else reject } y \end{aligned} \tag{3.206}$$

Figure 3.34 shows why this works in the case the majoring function is uniform. Each dot in this plot has coordinates

$$(y, u\,t(x)) \tag{3.207}$$

It makes intuitive sense that the values of y should be accepted based on the ratio between $t(x)$ and $f(x)$. In that way, the accepted values y will occur more frequently in the area where the ratio $t(x)/f(x)$ is close to unity.

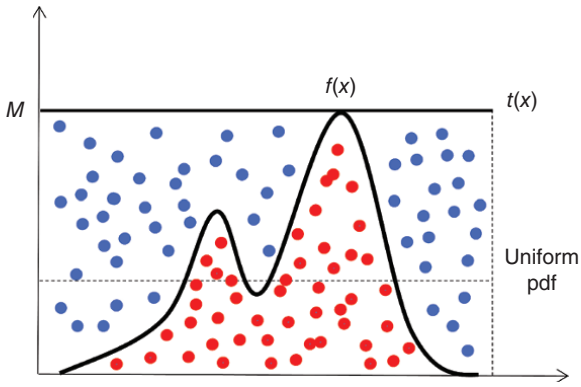


Figure 3.34 Sampling from a complex pdf by sampling from a uniform distribution and then accepting the red dots and rejecting the blue dots.

3.10.2.3. Compositions. Many applications of UQ in the subsurface require sampling from compositions. For example, lithologies such as shale and sand may exhibit different populations of petrophysical properties such as porosity and permeability. Hence, the total distribution of such properties may be expressed as

$$F(x) = \sum_{k=1}^{K} p_k F_k(x) \quad \text{with} \quad \sum_{k=1}^{K} p_k = 1 \;\text{ and }\; p_k \geq 0 \;\forall k \tag{3.208}$$

A mixture distribution with $K = \infty$ is termed a composition. Equation (3.208) can also be written in terms of densities. To sample from a mixture two draws are needed:

$$\begin{aligned} u_1 &\leftarrow \text{rand}(\bullet) \\ i &= \inf\left\{ k' : \sum_{k=1}^{k'} p_k \geq u_1 \right\} \\ u_2 &\leftarrow \text{rand}(\bullet) \\ x &= F_i^{-1}(u_2) \end{aligned} \tag{3.209}$$

3.10.2.4. Multivariate Gaussian. The Gaussian model is popular; hence, many ways exist to sample from it. The Gaussian model is applied in cases of not only multivariate modeling but also spatial modeling. In the latter case, the variables are the unknown random variables on a lattice or grid (random field). In the spatial case, one may already have measurements at certain locations (in geostatistics termed "hard data"). Without any hard data, the unconditional simulation of Gaussian field proceeds through, for example, an LU-decomposition (zero mean case):

$$C = LU \Rightarrow X = L^T \mathbf{y}, \quad \mathbf{y} \sim N(0,1) \tag{3.210}$$

C is the covariance matrix, which can be very large in the spatial case. To deal with these large matrices and also include the hard data, a sequential decomposition of the multivariate Gaussian into a chain of univariate Gaussians can be used:

$$\begin{aligned} f(X(\mathbf{s}_1),\ldots,X(\mathbf{s}_N)) = {} & f\left(X\left(\mathbf{s}_{(1)}\right)\right) \times f\left(X\left(\mathbf{s}_{(1)}\right)|X\left(\mathbf{s}_{(2)}\right)\right) \\ & \times f\left(X\left(\mathbf{s}_{(3)}\right)|X\left(\mathbf{s}_{(1)}\right)X\left(\mathbf{s}_{(2)}\right)\right) \times \cdots \\ & \times f\left(X\left(\mathbf{s}_{(N)}\right)|X\left(\mathbf{s}_{(N-1)}\right)\cdots X\left(\mathbf{s}_{(1)}\right)\right) \end{aligned} \tag{3.211}$$

where the notation (i) refers to a random permutation of the grid locations. Should some hard observation $\mathbf{d}_{\text{obs}}$ be available (and those are standard Gaussian or transformed to standard Gaussian), then

$$\begin{aligned} f(X(\mathbf{s}_1),\ldots,X(\mathbf{s}_N)|\mathbf{d}_{\text{obs}}) = {} & f\left(X\left(\mathbf{s}_{(1)}\right)|\mathbf{d}_{\text{obs}}\right) \\ & \times f\left(X\left(\mathbf{s}_{(1)}\right)|X\left(\mathbf{s}_{(2)}\right),\mathbf{d}_{\text{obs}}\right) \times \cdots \\ & \times f\left(X\left(\mathbf{s}_{(N)}\right)|X\left(\mathbf{s}_{(N-1)}\right)\cdots X\left(\mathbf{s}_{(1)}\right),\mathbf{d}_{\text{obs}}\right) \end{aligned} \tag{3.212}$$

A practical implementation of this is termed sequential Gaussian simulation [*Goovaerts*, 1997] and will be used in several applications later.

3.10.2.5. Using MC to Approximate a Distribution. MC arose as a technique to solve integrals. This looks perhaps a bit odd, since random number generators are used to solve a deterministic problem. However, calculating integrals numerically is not a trivial problem, in particular in higher dimensions. The number of samples drawn will then determine the accuracy of that solution. One particular integral problem is the definition of the expectation of some function

$$\mu_h = E[h(X)] = \int h(x)f(x)dx \quad X \sim f(x) \tag{3.213}$$

Using MC, this integral is estimated simply by the arithmetic mean from L samples $\{x^{(1)}, \ldots, x^{(L)}\}$ drawn from $f(x)$

$$\hat{\mu}_h = \frac{1}{L}\sum_{\ell=1}^{L} h\left(x^{(\ell)}\right) \tag{3.214}$$

In UQ, we are interested not only in just estimating a mean or a function but also in an uncertainty statement on some function value. Such uncertainty statement would need to involve a pdf or some quantiles calculated from that pdf. With a limited set of samples, this distribution can be represented by an empirical cdf:

$$\hat{F}(h(x)) = \frac{1}{L}\sum_{\ell=1}^{L} 1_{h(x^{(\ell)}) \le h(x)} \tag{3.215}$$

While this cdf is discontinuous, a continuous approximation can be obtained by making suitable interpolations and extrapolations [*Deutsch and Journel*, 1992]. In similar vein, the quantiles for given percentile $0 < p < 1$ can be estimated as

$$\hat{q}_p = \inf\left\{h(x) : \hat{F}(x) \ge p\right\} \tag{3.216}$$

Generally, the estimate $\hat{F}$ is unbiased but $\hat{q}_p$ is biased because it relies on the estimate $\hat{F}$ and not the true F.

3.10.3. Variance Reduction Methods

3.10.3.1. Introduction. In estimating or approximating integrals such as Eq. (3.202) or approximating a distribution using MC sampling, accuracy in the estimates is increased as the number of samples increases. However, evaluating the function $h(x)$ in Eq. (3.214) (such as a forward model) may be CPU demanding; hence, the number of MC samples must be limited. This calls for the definition of a measure of accuracy in the estimates. One such measure of accuracy is the variance of the estimates which measures the degree of error caused by limited sampling. A high variance in the estimates means low accuracy and a low variance of the estimates is a sign of high accuracy.

When performing simple also termed *naïve Monte Carlo*, we know that the sampling variance of Eq. (3.214) (a measure of degree of error caused by limited sampling) is given by

$$\text{var}(\hat{\mu}_h) = \frac{\sigma^2}{L}, \quad \sigma^2 = \text{var}(h(X)) \tag{3.217}$$

This estimate is also unbiased by definition (see Section 3.13 for a formal definition of bias). In case of biased estimators, a better measure of accuracy is the mean squared error (MSE) which is defined as

$$\text{MSE}(\hat{\mu}_h) = \text{var}(\hat{\mu}_h) + \text{bias}^2(\hat{\mu}_h) \tag{3.218}$$

Variance reduction aims to improve on the variance of Eq. (3.217) produced by the (naïve) MC sampling. Most variance reduction sampling schemes achieve this by sampling X more strategically. Random sampling may cause accidental clustering of samples $x^{(\ell)}$ in the space defined by X; hence, such samples are "wasted." Stratification (spreading the samples) is one such method. Another approach is to oversample (bias) certain regions that affect the calculation of the integrals most. Some weighting scheme must then be introduced to account for the incurred bias. This method, known as importance sampling, can be very efficient in reducing variance, but its improper use may lead to such sampling to run astray and actually increase variance. In other words, efficiency is traded off for the risk to have even less efficiency than the naïve sampler.

3.10.3.2. Stratified Sampling. The idea behind stratification is simple: split the high-dimensional model space of $\mathbf{X}$ into mutually exclusive and exhaustive regions and spread samples equally over these regions (instead of random of the entire domain as in the naïve MC), see Figure 3.35. Consider V the domain in which $\mathbf{X}$ varies and consider a partition of that domain into regions V_k: $k = 1, \ldots, K$. Denote

$$p_k = P(\mathbf{X} \in V_k) \tag{3.219}$$

Then, the conditional density of $\boldsymbol{X} \mid \boldsymbol{X} \in V_k$ is

$$f(\mathbf{x}|\mathbf{x} \in V_k) = \frac{1}{p_k} f(\mathbf{x}) 1_{\mathbf{x} \in V_k} \tag{3.220}$$

Stratified sampling becomes practical when we know the size of each strata V_k and we know how to sample from $f(\mathbf{x}|\, \mathbf{x} \in V_k)$; the latter is usually the case when the components of $\mathbf{X}$ are independent or when $\mathbf{X}$ follows a multi-Gaussian distribution. For example, when the components of $\mathbf{X} = (X_1, \ldots, X_N)$ are independent, then we sample a number of samples $L_k : k = 1, \ldots, K$, per region, preferably $L_k \geq 2$ (to estimate sampling variance). The samples per each region are denoted as $\left(\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(L_k)}\right)$. The stratified (and unbiased) estimate of Eq. (3.214) becomes

$$\hat{\mu}_{h,\text{strat}} = \sum_{k=1}^{K} \frac{p_k}{L_k} \sum_{\ell=1}^{L_k} h\left(\mathbf{x}_k^{(\ell)}\right) \tag{3.221}$$

In the case the L_k are chosen proportional, namely $L_k = p_k L$ with L the total number of samples, then

$$\hat{\mu}_{h,\text{strat}} = \frac{1}{L} \sum_{k=1}^{K} \sum_{\ell=1}^{L_k} h\left(\mathbf{x}_k^{(\ell)}\right) \tag{3.222}$$

The variance of this estimate is

$$\text{var}\left(\hat{\mu}_{h,\text{strat}}\right) = \sum_{k=1}^{K} p_k^2 \frac{\sigma_k^2}{L_k} \quad \text{with} \quad \sigma_k^2 = \text{var}(\mathbf{X}|\mathbf{X} \in V_k) \tag{3.223}$$

From this it becomes clear that a good stratification scheme has small within strata variances. The optimal allocation of L_k depends on the sampling cost per each stratum. If these are the same then the optimal L_k is given as

$$L_k = \frac{L p_k \sigma_k}{\sum_{k'=1}^{K} L p_{k'} \sigma_{k'}} \tag{3.224}$$

In actual applications, however, the σ_k may not be known (a priori) or may have to be estimated. Under such circumstances a safe bet is to use proportional allocation. Because high-dimensional space becomes empty very rapidly, stratification becomes difficult to implement beyond dimension larger than 5. As we will see in Chapter 8, it is most useful when sampling a subset of the variables within a larger MC sampling scheme involving other methods of variance reduction such as importance sampling.

3.10.3.3. Latin Hypercube Sampling. The main issue with stratification is that a space becomes quite empty for dimensions larger than 5. For example, in any dimension N, a regular grid has only $L^{1/N} \ll L$ strata per each component of the random vector (of dimension N) with L samples.

In Latin hypercube sampling [LHS; *Mckay*, 1998], we consider more than one dimension at a time to avoid the $L^{1/N}$ problem. The method is quite straightforward and easily explained when $d = 2$ (see Figure 3.36). Instead of assigning a number of samples per each square, one now assigns a sample per each row *and* column combined.

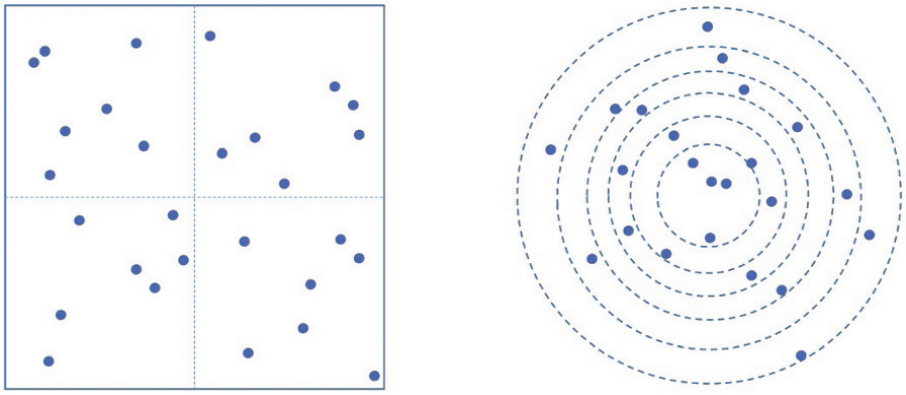


Figure 3.35 Sampling from a uniform distribution with seven samples per stratum. Sampling from a Gaussian distribution with four samples per stratum constructed from concentric circles.

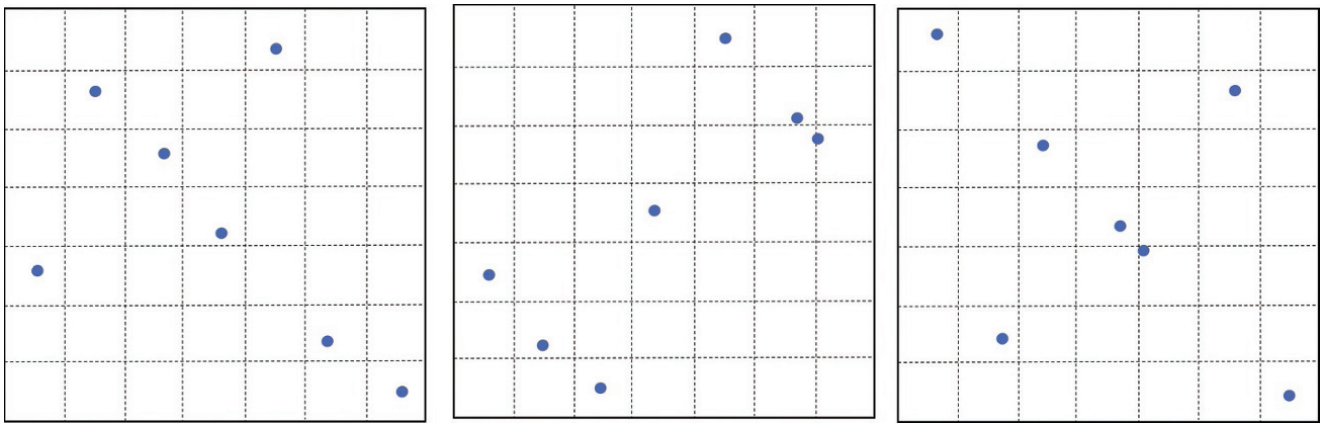


Figure 3.36 Three Latin hypercube samples in two dimensions.

The same idea is easily extended to higher dimensions. Note that in this context, the maximum number of combinations for a LHS with K divisions and N variables is $(K!)^{N-1}$. For example, a LHS with $K=4$ divisions and dimension of $\boldsymbol{X}$ equal to 3 will have only 576 possible combinations of samples.

Theory shows that LHS can be much better than naïve MC, and moreover it is robust in the sense that the bound on the sampling variance is almost the naïve MC sampling variance for Eq. (3.214) (up to a factor of $n/(n-1)$ only). In other words, it cannot get worse and for most functions h performs much better.

3.10.3.4. Control Variates and Multilevel MC. In the method of control variates, the variance of the sampler is reduced (compare to naïve MC) by the introduction of another variable termed the control variable. The basic idea is for this additional variable to be correlated to the target, but that performing MC on this control variable has much less cost than for the target. Next to having samples of the target $h(\mathbf{x}_1)$, …, $h(\mathbf{x}_L)$ to compute the estimate Eq. (3.214), we now consider samples available of a control variable denoted by $(c_1, \ldots, c_L)$; hence, we have some estimate

$$\hat{\mu}_c = \frac{1}{L}\sum_{\ell=1}^{L} c_\ell \tag{3.225}$$

If $h(\boldsymbol{X})$ and C correlate, then this estimate will inform the estimate $\hat{\mu}_h$, whether it is too small or too large, for example by comparing $\hat{\mu}_c$ for a sample size of L with $\hat{\mu}_c$, assumed to be quite accurately known through extensive sample ($\gg L$). A control variate estimator is constructed as follows:

$$\hat{\mu}_{cv} = \frac{1}{L}\sum_{\ell=1}^{L}\left(h\left(\mathbf{x}^{(\ell)}\right) + \beta(\mu_c - c_\ell)\right) \tag{3.226}$$

A simple calculation shows that the optimal value of β, under the criterion of minimizing the estimation variance, is

$$\beta = \frac{\mathrm{cov}(h(\mathbf{X}), C)}{\mathrm{var}(C)} \tag{3.227}$$

which makes sense, since the better the correlation, the more the covariate approximates the unknown μ_h. The main problem is that estimating β requires knowing the mean μ_h, which defeats the very purpose of sampling. For that reason β is estimated using the sample $(h(\mathbf{x}^{(1)}), c_1)$, …, $(h(\mathbf{x}^{(L)}), c_L)$

$$\hat{\beta} = \frac{\sum_{\ell=1}^{L} h\left(\mathbf{x}^{(\ell)}\right)c_\ell + L\hat{\mu}_h\hat{\mu}_c}{(L-1)\,\sigma_c^2} \quad \sigma_c^2 = \frac{1}{L-1}\sum_{\ell=1}^{L}(c_\ell - \hat{\mu}_c)^2 \tag{3.228}$$

Multilevel MC builds further on the idea of covariates. In multilevel MC, as the term suggests, we use more than one covariate (two levels: the target and a covariate). The goal again is to estimate $E[h(\mathbf{X})]$ of which samples $h_1(\mathbf{x}^{(\ell)})$, $\ell = 1, \ldots, L_1$ ($h_1 = h$) are available as well as samples $h_0(\mathbf{x}^{(\ell)})$, $\ell = 1, \ldots, L_0$ whose sampling is much less costly than h_1 ($L_0 > L_1$), then

$$E[h(\mathbf{X})] = E[h_1(\mathbf{X})] = E[h_0(\mathbf{X})] + E[h_1(\mathbf{X}) - h_0(\mathbf{X})] \tag{3.229}$$

This method again relies on the difference between the approximation and the target, which is estimated by

$$\hat{\mu}_{h,2L} = \frac{1}{L_0}\sum_{\ell=1}^{L_0} h_0\left(\mathbf{x}^{(\ell)}\right) + \frac{1}{L_1}\sum_{\ell=1}^{L_1}\left(h_1\left(\mathbf{x}^{(\ell)}\right) - h_0\left(\mathbf{x}^{(\ell)}\right)\right) \tag{3.230}$$

$2L$ refers to a two-level MC. The difference now with covariates is that $\beta = 1$ and the expected value of h_0 needs

to be estimated (instead of assumed known with high accuracy). The question instead is what appropriate values for L_0 and L_1 are that minimize the estimation variance of $\hat{\mu}_{h,2L}$. To that end, consider the following costs and estimation variances:

$$\begin{aligned} &\text{cost}_0, \text{var}_0 : \text{the cost and variance for } h_0(\mathbf{X}) \\ &\text{cost}_1, \text{var}_1 : \text{the cost and variance for } h_1(\mathbf{X}) \end{aligned} \tag{3.231}$$

Then based on Eq. (3.230), the total variance is

$$\text{var}_{\text{total}} = \frac{\text{var}_0}{L_0} + \frac{\text{var}_1}{L_1} \tag{3.232}$$

which is minimized for some fixed total cost by choosing

$$\frac{L_1}{L_0} = \frac{\sqrt{\frac{\text{var}_1}{\text{cost}_1}}}{\sqrt{\frac{\text{var}_0}{\text{cost}_0}}} \quad \text{or} \quad L_1 \sim \sqrt{\frac{\text{var}_1}{\text{cost}_1}};\ L_0 \sim \sqrt{\frac{\text{var}_0}{\text{cost}_0}} \tag{3.233}$$

This makes sense, since one would like for each level (target and covariate) to run an amount of simulations that is proportional to the accuracy (estimation variance) and inversely proportional to the cost of doing so. The two-level MC can be extended to a multilevel MC with various covariates with different approximations of the target. Consider these levels as 0, 1, …, M with $h_M = h$ the target, then Eq. (3.230) can be generalized to

$$\begin{aligned} \hat{\mu}_{h,M} = &\frac{1}{L_0}\sum_{\ell=1}^{L_0} h_0\left(\mathbf{x}^{(\ell)}\right) \\ &+ \sum_{m=1}^{M} \frac{1}{L_m}\sum_{\ell=1}^{L_m}\left(h_m\left(\mathbf{x}^{(\ell)}\right) - h_{m-1}\left(\mathbf{x}^{(\ell)}\right)\right) \end{aligned} \tag{3.234}$$

Again, one chooses an amount of simulations based on cost and estimation variance

$$L_m \sim \sqrt{\frac{\text{var}_m}{\text{cost}_m}} m = 0, \ldots, M \tag{3.235}$$

3.10.3.5. Importance Sampling

3.10.3.5.1. Methodology. The previous two methods used the idea of correlated samples to steer sampling of $f(\mathbf{x})$ to areas of the sampling domain that lead most to variance reduction of a particular estimate, without inducing (much) bias. The importance sampler uses a different strategy. It still tries to steer the sampler to important areas of the domain but does so by means of another pdf $q(\mathbf{x})$ (instead of a covariate).

Because sampling is done from the wrong distribution, a correction will need to be made to obtain unbiased estimates. In that sense, importance sampling is more than a variance reduction technique of an estimate It can also be used to perform MC on a given distribution by means of sampling from another distribution. Since UQ relies on some form of MC, we will see that importance sampling has many applications in UQ (see Chapter 7). For that reason, it can also be seen as an alternative to rejection sampling and lies at the foundation of sequential MC, which is treated in the next section.

Consider again the estimation problem of (3.214) with samples distributed as $f(\mathbf{x})$. Consider now another distribution $q(\mathbf{x})$ and write the expectation of h as follows:

$$\mu_h = \int_V h(\mathbf{x}) q(\mathbf{x}) \frac{f(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \tag{3.236}$$

One notices how the portion $h(\mathbf{x})q(\mathbf{x})$ in the integral leads to calculation of the expected value of $h(\mathbf{x})$ under $q(\mathbf{x})$, which is then corrected by a ratio of two pdfs $f(\mathbf{x})$ and $q(\mathbf{x})$. In importance sampling, we generate samples $\mathbf{x}^{(\ell)}$ from $q(\mathbf{x})$ and not $f(\mathbf{x})$, which then produces the estimate

$$\hat{\mu}_h = \frac{1}{L}\sum_{\ell=1}^{L} h\left(\mathbf{x}^{(\ell)}\right)\frac{f\left(\mathbf{x}^{(\ell)}\right)}{q\left(\mathbf{x}^{(\ell)}\right)} = \frac{1}{L}\sum_{\ell=1}^{L} w_\ell h\left(\mathbf{x}^{(\ell)}\right) \tag{3.237}$$

which requires the calculation of the ratio f/q. A sufficient condition for q is

$$f(E) = 0 \text{ such that } q(E) = 0 \text{ for that set } E \tag{3.238}$$

This basically entails that q must "cover" the range of f, being able to generate samples wherever f generates samples. The most critical question evidently is on good choices for q. Statistical theory provides some suggestions [*Lemieux*, 1997] based on general principles, but having domain knowledge (the specific application) on the target distribution may be more important. For example, where sampling of f has greatest impact on UQ and perhaps even the decisions made is probably more relevant to the context of this book. In that respect, any sampling to estimate Eq. (3.214) can also be used to approximate a distribution. Nevertheless, the choice of q will determine the amount of variance reduction achieved. Chapter 7 uses such domain knowledge to decide on a proposal distribution.

Consider a simple example in Figure 3.37 where the goal is to estimate $P(X > 4)$ where $X \sim N(0, 1)$ and hence the true exceedance probability $p_{\text{true}} = 3.1671 \times 10^{-5}$. When performing naïve MC with 100.000 samples, we get the estimate $\hat{p}_{\text{naive}} = 3.5000 \times 10^{-5}$. Consider now another pdf more centered around the target area of interest, $q(\mathbf{x}) \sim N(4, 2)$. When applying Eq. (3.237) we find that $\hat{p}_{\text{imp}} = 3.1676 \times 10^{-5}$.

3.10.3.5.2. Guideline for q. To study the sampling properties of importance sampling, and the impact of certain choices of q, recall that the variance of the MC sampler is

$$\text{var}\left(\hat{\mu}_{h,MC}\right) = \frac{1}{L}\left(E\left[h^2(\mathbf{X})\right] - \mu_h^2\right) \quad \mathbf{X} \sim f(\mathbf{x}) \tag{3.239}$$

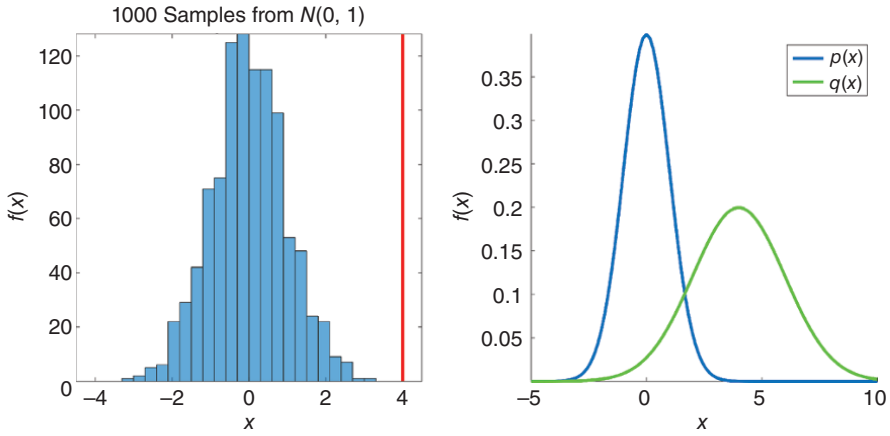


Figure 3.37 Using MC to estimate an exceedance probability from the blue pdf. Thousand samples provide basically no accuracy, while high accuracy can be obtained by using the green proposal distribution.

A simple calculation shows that

$$\mathrm{var}\left(\hat{\mu}_{h,MC}\right) = \frac{1}{L}\left(E\left[h^2(\mathbf{X})\frac{f(\mathbf{X})}{q(\mathbf{X})}\right] - \mu_h^2\right) \quad \mathbf{X} \sim f(\mathbf{x}) \tag{3.240}$$

Hence, importance sampling is more efficient than MC when

$$E\left[h^2(\mathbf{X})\frac{f(\mathbf{X})}{q(\mathbf{X})}\right] \le E\left[h^2(\mathbf{X})\right] \tag{3.241}$$

As a result, to be efficient, the ratio f/q should be small, making $\mathbf{x}$ more likely (according to that ratio) when $h(\mathbf{x})$ is larger. When $h(\mathbf{x})$ is small, then one should aim for a ratio larger than unity. Note that only when

$$\frac{f(\mathbf{x})}{q(\mathbf{x})} < 1 \quad \forall \mathbf{x} \quad \text{for which} \quad h(\mathbf{x}) \neq 0 \tag{3.242}$$

leads to variance reduction. This is also means that a poorly chosen q will lead to a variance increase. The more one knows about $h(\mathbf{x})$ and $f(\mathbf{x})$ the better a choice for q can be made, but again, this is very application specific. Getting insight into $g(\mathbf{x})$ and $f(\mathbf{x})$ in very high dimensions is not trivial. We will show in Chapter 7 that dimension reduction methods become very helpful in this regard.

q cannot be light-tailed compared to f, but a heavy tailed q often leads to inefficient sampling (close to rejection sampling). To study more quantitatively the importance sampling weights, a so-called effective sample size is introduced. When all weights are equal then the effective sample size is basically L. However, when weights become skew, then fewer samples have influence on the estimate of h. Denote as w_ℓ the weight given to a sample of q, namely $\mathbf{x}^{(\ell)}$, then an effective sample size L_{eff} can be defined as

$$L_{\text{eff}} = L\frac{\sum_{\ell=1}^{L} w_\ell}{\sum_{\ell=1}^{L} w_\ell^2} \tag{3.243}$$

Clearly, $L_{\text{eff}} \ll L$ when the weights are very skew. If L_{eff} is very small, then the importance sampler estimate may not be trusted and, worse, it may lead to an increase in variance. In the next section, on sequential MC, we will see how this problem can be alleviated.

3.11. SEQUENTIAL MC

3.11.1. Problem Formulation

Previously, we discussed methods to sample from univariate or higher-dimensional distributions. Here we consider a more specific problem, namely sampling from higher-dimensional distributions conditioned on observations. In particular, predicting, by means of MC, a future "signal" (realization, unknown, sample, event) from a past "signal" (typically denoted as "data"). In addition, this problem is formulated dynamically in time, not for a single static instance. The idea is to "assimilate" data as time progresses to make forecasts on some target future variable or event. Future observations will then become data as time progresses. Forecasting weather is an evident example [*Leeuwen and Jan*, 2009], but many other applications exists such as robotic navigation [*Dellaert et al.*, 2001], financial market analysis [*Aihara et al.*, 2009], visual object tracking [*Nummiaro et al.*, 2003], and so on. Data assimilation, data filtering, sequential MC (SMC), bootstrap filtering, particle filtering, and survival of the fittest sampling are all nomenclature to address basically the same problem.

Generally, SMC aims to dynamically predict an "unknown signal" from "observed data." Within the context of UQ, the data are modeled using data variables, but now occur at some discrete time events:

$$\mathbf{d}_{1:t} = \{\mathbf{d}_1, \ldots, \mathbf{d}_{t-1}, \mathbf{d}_t\} \quad t \in N^+ \tag{3.244}$$

For example, one may repeat 3D seismic surveys to obtain a sequence termed 4D seismic (each 3D is compared with the base survey) or, in a groundwater setting, one may repeat hydraulic head or concentration measurements over time. The "unknown signal" is the unknown subsurface, modeled using model variables; here these model variables are updated in time (no longer a single static model)

$$\mathbf{m}_{1:t} = \{\mathbf{m}_1, \ldots, \mathbf{m}_{t-1}, \mathbf{m}_t\} \tag{3.245}$$

Forward models are used to produce predictions

$$\mathbf{h}_{1:t} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{t-1}, \mathbf{h}_t\} \quad \mathbf{h}_{t'} = g_h(\mathbf{m}_{t'}) \tag{3.246}$$

Note that the forward model g_h is now explicitly written as a function of the time at which a prediction is made. Prior to having any data, hence at time $t = 0$, the initial or prior uncertainty on model and prediction are

$$\mathbf{m}_0 \sim f(\mathbf{m}_0); \quad \mathbf{h}_0 \sim f(\mathbf{h}_0) \quad (\text{no data yet}) \tag{3.247}$$

In addition, and this is an assumption used in most SMC, a Markov property is invoked whereby the unobserved signal is conditionally independent of all previous states $t-2$, …, 1 given state $t-1$. This basically means that knowing the previous state at $t-1$ is sufficient, and we do not need to include all the prior states $t-2$, …, 1. The probabilistic model is now fully specified by

$$\begin{aligned} &\text{Prior } f(\mathbf{m}_0) \\ &\text{Conditional}: f(\mathbf{m}_t|\mathbf{m}_{t-1}, \mathbf{m}_{t-2}, \ldots, \mathbf{m}_1) \cong f(\mathbf{m}_t|\mathbf{m}_{t-1}) \\ &\text{Conditional}: f(\mathbf{d}_t|\mathbf{m}_t, \mathbf{m}_{t-1}, \ldots, \mathbf{m}_1) \cong f(\mathbf{d}_t|\mathbf{m}_t) \end{aligned} \tag{3.248}$$

The aim is to recursively estimate the following:

1. $f(\mathbf{m}_{0:t}|\mathbf{d}_{1:t})$: the full posterior distribution of all models, in time, given the time sequence of data.
2. $f(\mathbf{m}_t|\mathbf{d}_{1:t})$: the marginal for the present model given past data, also termed the filtering distribution.
3. $\int h(\mathbf{m}_{0:t}) f(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) d\mathbf{m}_{0:t}$: any expectation of some function of interest. This could be a mean, a covariance but also some summary statistic of the model, such as the prediction h deduced from it. More specifically, the marginal $\int h(\mathbf{m}_t) f(\mathbf{m}_t|\mathbf{d}_{1:t}) d\mathbf{m}_t$.

In terms of the latter prediction, one could also aim to directly obtain $f(\mathbf{h}_{0:t}|\mathbf{d}_{1:t})$: the model is now a hidden variable.

Ways of obtaining samples from Eq. (3.248) will be the topic of Chapter 7. The solution to the above-formulated problem is obtained by the direct application of Bayes' rule:

$$f(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) = \frac{f(\mathbf{d}_{1:t}|\mathbf{m}_{0:t}) f(\mathbf{m}_{0:t})}{\int f(\mathbf{d}_{1:t}|\mathbf{m}_{0:t}) f(\mathbf{m}_{0:t}) d\mathbf{m}_{0:t}} \tag{3.249}$$

Then, a recursive formula going from time t to $t+1$ is obtained as

$$f(\mathbf{m}_{0:t+1}|\mathbf{d}_{1:t+1}) = f(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) \frac{f(\mathbf{d}_{t+1}|\mathbf{m}_{t+1}) f(\mathbf{m}_{t+1}|\mathbf{m}_t)}{f(\mathbf{d}_{t+1}|\mathbf{d}_{1:t})} \tag{3.250}$$

In terms of the marginal, one therefore obtains the following recursion:

$$\begin{aligned} &\text{Predict } t \text{ from } 1:t-1: \; f(\mathbf{m}_t|\mathbf{d}_{1:t-1}) \\ &\quad = \int f(\mathbf{m}_t|\mathbf{m}_{t-1}) f(\mathbf{m}_{t-1}|\mathbf{d}_{1:t-1}) d\mathbf{m}_{t-1} \\ &\text{Update/assimilate data } \mathbf{d}_t: \; f(\mathbf{m}_t|\mathbf{d}_{1:t}) \\ &\quad = \frac{f(\mathbf{m}_t|\mathbf{d}_t) f(\mathbf{m}_t|\mathbf{d}_{1:t-1}) d\mathbf{m}_t}{\int f(\mathbf{m}_t|\mathbf{d}_t) f(\mathbf{m}_t|\mathbf{d}_{1:t-1}) d\mathbf{m}_t} \end{aligned} \tag{3.251}$$

This iterative/recursive of updating models with data requires the solution of integrals over high-dimensional functions. From our previous discussion, see Section 3.10.2.5, we saw how MC can be used to estimate these integrals. This would be "perfect MC" sampling. However, such MC would be very inefficient, in particular if evaluation of either data forward models or prediction forward models are CPU demanding. Perfect MC ignores the sequential nature of these integrals (the next one depends of the previous one only because of the Markov property) and thereby ignoring important information to make such calculation more efficient. How this is achieved is discussed in the following section on a sequential sampling method termed SIR (sequential importance resampling).

3.11.2. Sequential Importance Resampling

Recall that importance sampling aims to calculate an expectation more accurately with lesser samples than a naïve MC. This was achieved by sampling from another distribution, then reweighting the samples to correct for that biased sampling. Importance sampling can equally be applied as a variance reduction method within sequential MC. In this context and using the notation in the context of the importance sampling method, see Eq. (3.237), we substitute as follows:

$$E[h(\mathbf{X})] \to E[h(\mathbf{m}_{0:t})] \tag{3.252}$$

$$f(\mathbf{x}) \to f(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) \tag{3.253}$$

$$q(\mathbf{x}) \to q(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) \tag{3.254}$$

$$\mathbf{x}^{(\ell)} \rightarrow \mathbf{m}^{(\ell)} \tag{3.255}$$

$$\Rightarrow \hat{\mu}_{h(\mathbf{m}_{0:t}),IS} = \frac{1}{L}\sum_{\ell=1}^{L} w\left(\mathbf{m}_{0:t}^{(\ell)}\right) h\left(\mathbf{m}_{0:t}^{(\ell)}\right); \quad w\left(\mathbf{m}_{0:t}^{(\ell)}\right) = \frac{f(\mathbf{m}_{0:t}|\mathbf{d}_{1:t})}{q(\mathbf{m}_{0:t}|\mathbf{d}_{1:t})} \tag{3.256}$$

which is estimated from L samples

$$\left\{\mathbf{m}_{0:t}^{(1)},\ldots,\mathbf{m}_{0:t}^{(L)}\right\} \tag{3.257}$$

These samples are also termed "particles": think of points floating around in a very high dimensional space. The problem with this simple application of importance sampling within the context SMC is that the weights need to recalculated at each time t, since models get updates and hence weights need updating, regardless of how suitable a choice for q is taken. As time progresses, this becomes progressively more difficult from a computational point of view.

One way around this is to evoke again the Markov assumption and state that q at time t is conditionally depended on q at time $t-1$

$$q(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) = q(\mathbf{m}_{0:t-1}|\mathbf{d}_{1:t-1})q(\mathbf{m}_t|\mathbf{m}_{0:t-1},\mathbf{d}_{1:t}) \tag{3.258}$$

For example, if q is simply the prior distribution $f(\mathbf{m})$ (which will certainly cover the target posterior distributions), then

$$q(\mathbf{m}_{0:t}|\mathbf{d}_{1:t}) = f(\mathbf{m}_0)\prod_{t'=1}^{t} f(\mathbf{m}_{t'}|\mathbf{m}_{t'-1},\mathbf{d}_t) \tag{3.259}$$

The problem, however, with this approach is that as t progresses, the weights become very skewed, meaning that only very few samples (particles) get any weight (also termed degeneracy). This makes intuitive sense: as data becomes more constraining, the posterior distribution becomes narrower, and given a limited set of prior models, few of those prior models fall within the range of the posterior distribution. We discussed in Section 3.10.3.5 that such skewness is not desirable, as IS estimates may have very large variance, even larger than the MC sampler.

SIR, also termed bootstrap filtering, allows dealing with this problem by eliminating from the set of prior models/particles those models that have low weights in the importance sampler and targeting the generation of particles/models with higher weight. SIR employs a simple resampling idea. Since importance weights are normalized ($w \geq 0; \sum w = 1$), these weights form a discrete empirical distribution (see also Section 3.10.3.5) on the current particles/models. We can, therefore, replace the weighted estimate of Eq. (3.256) into an unweighted estimate by resampling particles/models from this discrete distribution

$$\hat{\mu}_{h,\mathrm{SIR}} = \frac{1}{L}\sum_{\ell=1}^{L} h\left(\tilde{\mathbf{m}}_{0:t}^{(\ell)}\right) \tag{3.260}$$

with $\tilde{\mathbf{m}}_{0:t}^{(\ell)}$ sampled according to the importance sampler weights $\mathbf{w}_{0:t}$. The advantage of SIR is that (i) it is easy to implement because it is very modular (generate model, calculate weights, resample models), hence no iterative inversion is needed and (ii) it is perfectly parallelizable since the forward models can be applied to all samples at the same time.

As additional observations become available, $\mathbf{d}_{t+1}$, the process is repeated. The proposal function is reestimated using the resampled particles from the previous time step, and Eq. (3.258) is rewritten as

$$q(\mathbf{m}_{0:t+1}|\mathbf{d}_{1:t+1}) = q(\mathbf{m}_{0:t}|\mathbf{d}_{1:t})q(\mathbf{m}_{t+1}|\mathbf{m}_{0:t},\mathbf{d}_{1:t+1}) \tag{3.261}$$

This process is repeated each time new observations are gathered, providing an online estimate of $\mathbf{m}_t$ at any subsequent time step.

3.12. MARKOV CHAIN MC

3.12.1. Motivation

Markov chain MC (McMC) methods [*Geyer*, 2002] are used to sample iteratively from complicated distributions for which MC methods do not work anymore. Such situations arise, for example in Chapter 6 on inversion, where the aim is to infer model variables from data. While McMC methods apply to sampling from any distribution model, the interest in this book mostly lies in sampling from a posterior model within a Bayesian context (see Chapters 5 and 6); hence, the distribution to be sampled from is defined as

$$f(\mathbf{m}|\mathbf{d}) = \frac{f(\mathbf{d}|\mathbf{m})}{f(\mathbf{d})} f(\mathbf{m}) \simeq f(\mathbf{d}|\mathbf{m}) f(\mathbf{m}) \tag{3.262}$$

A number of complications may arise when sampling from such posterior model:

1. The prior model $f(\mathbf{m})$ may not have an analytical expression, but rather is some computer algorithm that generates prior model realizations $\mathbf{m}^{(1)}$, $\mathbf{m}^{(2)}$, $\mathbf{m}^{(3)}$, …; hence, the density value $f(\mathbf{m}^{(\ell)})$ cannot be evaluated. This is not unusual in subsurface applications where complex computer codes generate very realistic representations of the subsurface. Basically this code maps random numbers into some subsurface model, without explicit knowledge of the underlying density.

2. The forward modeling code used to evaluate how well a model matches the data is again a complex computer algorithm to which we have little or no access.

The main point here is that in both (1) and (2) we have only access to the "black box" for generating prior models and for applying some response function on these prior model.

3.12.2. Random Walk

Since we cannot sample directly and independently from the posterior, we need to "walk around" the sample space iteratively and hope to find high-likelihood models. Jumping around randomly is not a good strategy. A better strategy would be to walk around models that have high likelihood, once we have found such a region. Unless the posterior is quite degenerate (random peaks), it is more plausible that higher likely models can be found in such region. This suggests walking around more like a drunk person (a random walk) than a grasshopper. The problem now is how to merge this random walking with sampling properly from a distribution. Intuitively, the random walk as a sampling method makes sense: walking around more frequently in high-likelihood areas than low-likelihood areas will generate samples from the posterior that reflects that frequency. The list of samples obtained along this random walk, however, does not constitute a sample from the posterior, because two consecutive samples along this random walk are not independent (and we would like i.i.d. samples). In fact, to make this walking relevant in terms of finding high-likelihood samples, they should be very dependent.

The random walk initiates a Markov chain of models $\mathbf{m}$ such that the next model is generated only knowing the current model. The key theoretical result is that if such chain of models is (i) irreducible and not transient, meaning essentially that any model can be "reached" from any other model through the walk, and (ii) is aperiodic (e.g., it does not get stuck in loops), then the Markov chain has a stationary distribution that equals the target distribution, here the posterior distribution. If this walking around is done "long enough" then the sample obtained when stopping the walk is a sample from the posterior distribution. More samples can be obtained by starting over, or waiting again long enough. These issues will be discussed in Section 3.12.5.

The McMC theory does, however, not state how such walking (iterating) should be performed but provides only the necessary and sufficient conditions for it. Hence, many methods have been designed to perform McMC sampling. Unless in very specific cases (e.g., Gaussian), there is also no theoretical result on when to stop (i.e., when "convergence" is reached). In that sense, McMC constitutes a family of approximate samplers.

3.12.3. Gibbs Sampling

The Gibbs sampler [*Geman and Geman*, 1984] is particularly useful for high-dimensional problems because it relies on dividing the model variables into "blocks":

$$\mathbf{m} = (\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K) \tag{3.263}$$

Each iteration of the Gibbs sampler consists of cycling through all these blocks $\mathbf{m}_k$ and drawing a new value for each block conditional on all the other blocks. These conditional distributions are, therefore, represented as

$$f(\mathbf{m}_k | \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{k-1}, \mathbf{m}_{k+1}, \ldots, \mathbf{m}_K, \mathbf{d}) \tag{3.264}$$

Notice how these conditional distributions are in fact partial posterior distribution in the model variables. Gibbs sampling is particularly useful in cases where the sequence of conditional distribution is available, such as in hierarchical Bayesian models [*Jaynes*, 2003], or in sequential simulation, often used in geostatistics [*Hansen et al.*, 2012].

3.12.4. Metropolis–Hastings Sampler

The Metropolis sampler [*Hastings*, 1970] adds an acceptance/rejection rule to the random walk and works as follows:

Generate an initial model from the prior: $\mathbf{m}^0 \sim f(\mathbf{m})$

For $k = 1, 2, \ldots$

1. Sample a new model from the proposal distribution $\mathbf{m}^* \sim f(\mathbf{m} | \mathbf{m}^{k-1})$

2. Calculate the ratio $\alpha = \frac{f(\mathbf{d}|\mathbf{m}^*)}{f(\mathbf{d}|\mathbf{m}^{k-1})}$

3. Set $\mathbf{m}^k = \begin{cases} \mathbf{m}^* & \text{with probability } \min(\alpha, 1) \\ \mathbf{m}^{k-1} & \text{else} \end{cases}$

For the Metropolis sampler, the proposal distribution must be symmetric $f(\mathbf{m}^i | \mathbf{m}^j) = f(\mathbf{m}^j | \mathbf{m}^i)$. The Metropolis–Hasting sampler, on the other hand, generalizes this to asymmetric proposal distributions, a much wider class of proposal distributions. Therefore, it requires the calculation of the following ratio which accounts for this asymmetry:

$$\alpha = \frac{f(\mathbf{d}|\mathbf{m}^*)}{f(\mathbf{d}|\mathbf{m}^{k-1})} \frac{f(\mathbf{m}^{k-1}|\mathbf{m}^*)}{f(\mathbf{m}^*|\mathbf{m}^{k-1})} \tag{3.265}$$

The use of an asymmetric proposal distribution often aids in increasing the speed of the random walk [*Gelman et al.*, 2004]. The ideal jumping distribution is the target distribution. In that case $\alpha = 1$, always; the sampler becomes a series of independent draws from the target distribution. This ideal case does not happen, but some desirable properties of jumping distributions are that (i) they are easy to sample from and (ii) they allow relatively large jumps and do not often get rejected. The Gibbs sampler and

Metropolis sampler are often used as basic building blocks in constructing more efficient samplers (see Section 3.12.6).

3.12.5. Assessing Convergence

MC by independent draws (i.i.d sampling) is simple: the resulting outcomes are samples from the target distribution (here the posterior distribution) and the empirical distribution approximates the target distribution. Sampling iteratively is more challenging because of the following reasons:

1. The iterations have to proceed long enough; otherwise, the sampler will reflect more the starting approximation rather than the actual distribution.

2. Since sampling is not done by independent draws, serial correlation exists between the samples that may cause inefficiency in the sampling. Even though at "convergence" (which is theoretically infinite but has good approximation for finite iterations), the correlated draws are samples from the target distribution, the amount of samples is effectively less than if one would sample by independent draws. The higher this correlation, the less the effective sample size is.

This requires methods for "monitoring" convergence. Note that such methods usually provide necessary indications/conditions that we are drawing from the target distribution, they are however not sufficient. The first stage of monitoring is with regard to the "burn-in" or "warm-up." Here we see significant changes in any properties calculated during in the chain. Any statistics that are being monitored display nonstationary variation. A common approach to monitoring is to start multiple chains and then study so-called "mixing" of the chains. In this mixing, we study two levels of variation in the chain: the variation of some quantity q (mean, variance, median, etc.) within a single chain and the variation between chains (similar to a variance analysis in cluster sampling). Consider again our target distribution $f(\mathbf{m}|\,\mathbf{d})$ and the monitoring of some estimated $\hat{q}$ (a scalar such as an estimate of the mean of the posterior distribution). Consider that we run m chains and that we are currently at iteration n. We compute

$$\begin{aligned} B_{\hat{q}} &: \text{the between variance of } \hat{q} \\ W_{\hat{q}} &: \text{the within variance of } \hat{q} \end{aligned} \tag{3.266}$$

$W_{\hat{q}}$ provides an estimate of the marginal posterior variance of q; however, this is an underestimate (has less variability) because the target distribution has not been fully sampled yet. One proposal, therefore, is to calculate a "corrected" estimate as

$$W_{\hat{q},\text{corr}} = W_{\hat{q}} + \frac{1}{n} B_{\hat{q}} \tag{3.267}$$

In the limit this will estimate the variance unbiasedly. In monitor convergence, we can therefore study the ratio

$$R = \frac{W_{\hat{q},\text{corr}}}{W_{\hat{q}}} \tag{3.268}$$

which should converge to 1 as $n \to \infty$.

3.12.6. Approximate Bayesian Computation

A "full" Bayesian method involves specifying models for the likelihood and prior and then to sample from the posterior. These models are full multivariate pdfs. In Chapter 6 we will discuss various applications of the full Bayesian approach. The problem with full Bayesian methods is that they often rely on certain model assumptions. One example is the overuse of the Gaussian distribution (by lack of anything better) or specification of likelihood by assuming independence in measurement errors. In that context, *Schoups and Vrugt* [2010] introduced a generalized likelihood function where residual errors are correlated, heteroscedastic, and non-Gaussian. However, within a geological context this may not be the main issue. Even if this allows for a more general likelihood function, it does not address how errors (model, data, epistemic, aleatory) can be separated. Full Bayes may require extensive computations to fully evaluate the likelihood model and/or normalization constant. An example of the latter is the unknown normalization constants in MRF models that may require McMC sampling just to evaluate this normalization constant [*Tjelmeland and Besag*, 1998; *Mariethoz and Caers*, 2015].

From a subsurface system application point of view, a full Bayesian approach may not be needed. Adhering rigorously to model specifications and performing rigorous sampling ignores the subjectivity and importance of the prior model. Why would one sample from a very subjectively chosen model very rigorously? This point will be strongly argued in Chapters 5, 6, and 7. The critical issue in UQ is the statement of a physically realistic and geologically plausible prior distribution. Hence, the main problem is not so much the sampling, but what exactly one is sampling from. We may not need to care too much about the "correct" posterior model and the "correct" sampling from it. Rather we would like for our posterior models to adhere to properties formulated in a physics-based prior and to reflect some field data.

A large family of methods that avoid likelihood pdfs, and also called "likelihood-free" methods [*Diggle and Graton*, 1984], are termed approximate Bayesian computation (ABC). This simple idea has many varieties, see ABCs method in *Beaumont et al.* [2002], *Sadegh and Vrugt* [2014], *Sadegh and Vrugt* [2013], and *Turner and Van Zandt* [2012], or extended rejection sampling in

Mosegaard [1995], or generalized likelihood uncertainty estimation in *Beven* [2009]. The basic method is as follows:

1. Draw $\mathbf{m}$ from the prior distribution $f(\mathbf{m})$, simulate the data from the forward model $\mathbf{d} = g_d(\mathbf{m})$.
2. Specify a distance and calculate $d(\mathbf{d}, \mathbf{d}_{\text{obs}})$.
3. Accept $\mathbf{m}$ if $d(\mathbf{d}, \mathbf{d}_{\text{obs}}) < \varepsilon$ for some specified threshold ε, otherwise reject.

This method accepts draws from the prior relative to the distance calculated from the observed data. There are three issues that need to be addressed: (i) specify an appropriate distance, note that $\mathbf{d}$ may be very high dimensional, hence comparison will be made based on some summary statistics evaluated on $\mathbf{d}$, (ii) ε should be small, but not too small, and (iii) for certain prior models, the rejection rate may be very high.

What approximation is being made compared to a full Bayesian approach? Given the above distribution, the approximate distribution being sampled from is

$$f(\mathbf{m}|\mathbf{d}) = f(\mathbf{m}) \int_d f(\mathbf{d}|\mathbf{m}) I(d(g_d(\mathbf{m}), \mathbf{d}_{\text{obs}}) < \varepsilon) d\mathbf{d} \quad (3.269)$$

with I an indicator operator (either 1 or 0). The actual (true) posterior distribution is obtained when $\varepsilon \to 0$ [*Beaumont et al*., 2002; *Turner and Van Zandt*, 2012]. In the above algorithm

$$f(\mathbf{d}|\mathbf{m}) = \delta(\mathbf{d} - g(\mathbf{m})) \quad (3.270)$$

but could also be based on a measurement error model (see Chapter 6)

$$\mathbf{d} = g(\mathbf{m}) + \varepsilon \quad (3.271)$$

3.12.7. Multichain McMC

Traditional McMC methods work well for problems that are not too high dimensional (a few parameters, e.g., five or less). For high-dimensional and nonlinear problems, with multimodel distributions in such spaces, these methods can become difficult to apply, or have problems in terms of efficiency and convergence. One way to overcome these inefficiency problems is to create so-called adaptive chains, meaning that the proposal distribution changes during iteration.

The most common adaptive single chain methods are adaptive proposal [*Haario et al*., 1999] and adaptive Metropolis [*Haario et al*., 2001] methods. The proposal distribution here is multivariate Gaussian and the adaptation is made in the covariance matrix, by recalculating the covariance based on a set of samples along the chain as well as some consideration on the dimensionality of the problem. Although increase in efficiency is achieved in higher dimensions, the nature of the adaption makes it really only applicable for Gaussian-type distributions.

Multiple chain methods use multiple chains running in parallel and are known to out-perform single chain methods for complex posterior distributions, for example, that exhibit multimodality (e.g., [*Gilks et al*., 1994; *Liu et al*., 2000; *Craiu et al*., 2009]). A multichain method popular in hydrology (as well applied to a variety of other problems) is the DiffeRential Evolution Adaptive Metropolis method [DREAM, *Vrugt et al*., 2009; *Gupta et al*., 2012; *Laloy and Vrugt*, 2012; *Vrugt*, 2016]. This method is based on an adaptive Metropolis sampler termed Differential Evolution Markov chain (DE-MC). DE-MC employs genetic algorithms (or any other differential evolution, [*Storn and Price*, 1997] to evolve a population of chains but using the Metropolis criterion (Eq. (3.265)) to evolve the population of chains [*Ter Braak*, 2006]. A traditional genetic algorithm (GA), (on its own) is not a sampler but an optimizer; hence, convergence is increased by combining GA and McMC, while still drawing from the posterior distribution. In DREAM, DE-MC is enhanced by using an adaptive randomized subspace sampling as well as other methods to get balance and ergodicity in the chain, leading to considerable improvement over other adaptive MCMC sampling approaches [*Vrugt*, 2016].

Another approach is to combine McMC with sequential MC. For example, *Andrieu et al*. [2010] propose a particle Markov chain Monte Carlo (PMCMC) methods, which relies on a combination of both McMC and SMC taking advantage of the strength of each. Here SMC algorithms are used to design efficient high-dimensional proposal distributions for MCMC algorithms.

3.13. THE BOOTSTRAP

3.13.1. Introduction

Even with the advent of new data scientific approaches such as machine learning, or computer vision, the basic approach to data analytics has not changed: (i) collect data, (ii) summarize data, and (iii) infer from data: statistical inference. In that regard, the bootstrap caused a revolution in statistical science since its inception [*Efron*, 1979; *Efron and Tibshirani*, 1994]. Statistical inference deals with the estimation of (population) parameter θ, in terms of estimates $\hat{\theta}$ and determining how accurate $\hat{\theta}$ is in terms of the true θ. A typical example is the estimate of the mean $\mu = E[X]$ of a random variable $X \sim F(x, \theta)$. An unbiased estimate of this expectation is the arithmetic average

$$\hat{\mu} = \frac{1}{L} \sum_{\ell=1}^{L} x^{(\ell)} \quad (3.272)$$

If the population variance σ is known, and $X \sim N(\mu, \sigma)$, then the sampling distribution of $\hat{\mu}$ (meaning how $\hat{\mu}$ varies in the population for that sample size) is

$$X \sim N\left(\hat{\mu}, \frac{\sigma}{\sqrt{L}}\right) \tag{3.273}$$

In case σ is not known, the standard error of the estimate can be calculated as

$$se_{\hat{F}} = \frac{s}{\sqrt{L}} \quad s = \sqrt{\frac{\sum_{\ell=1}^{L}\left(x^{(\ell)} - \hat{\mu}\right)^2}{L-1}} \tag{3.274}$$

The problem is that this simple setting does not easily extend to the general case for any random variable and any quantity of interest that needs to be estimated. Explicit expressions are usually not available.

The bootstrap instead provides an approximation of $se_{\hat{F}}$ by way of resampling. To that extent, we introduce the notion of a bootstrap sampling which samples from the original sample $(x^{(1)}, \ldots, x^{(L)})$ with replacement (basically putting all samples in a bag, taking a value, but putting it back too). We denote such sample as

$$\mathbf{x}_b = \left(x_b^{(1)}, \ldots, x_b^{(L)}\right) \tag{3.275}$$

The value $x_b^{(\ell)}$ can only be one of the original sample values. This resampling idea allows for values to be sampled multiple times, even if they are only found once in the original sample. Essentially, samples are drawn from the (discrete) empirical distribution constructed from the original sample. As many such new sample sets can be generated as desired

$$\mathbf{x}_b = \left(x_b^{(1)}, \ldots, x_b^{(L)}\right) \quad b = 1, \ldots, B \tag{3.276}$$

Using each such bootstrap sample, we recalculate the estimate as $\hat{\hat{\theta}}_b$ (the double hat indicating it is calculated under the bootstrap, the subscript, which bootstrap sample) and calculate the bootstrap standard error as

$$\widehat{se}_B\left(\hat{\theta}\right) = \sqrt{\frac{\sum_{b=1}^{B}\left(\hat{\hat{\theta}}_b - \hat{\hat{\theta}}\right)^2}{B-1}} \quad \hat{\hat{\theta}} = \frac{\sum_{b=1}^{B}\hat{\hat{\theta}}_b}{B} \tag{3.277}$$

Note, however, that θ does not have to be the mean, it could be any parameter. Figure 3.38 provides a summary of the procedure.

3.13.2. Nonparametric Bootstrap

3.13.2.1. One-Sample. In this section, we focus on the simple case of a single sample of a random variable $(x^{(1)}, \ldots, x^{(L)}) \sim F(x, \theta)$. This sample generates an empirical distribution $\hat{F}$ with emprical probability of $1/L$ on each sample $x^{(\ell)}$. The true parameter of a distribution can be written as a general function (e.g., an integral) of the true distribution

$$\theta = t(F) \tag{3.278}$$

Using the empirical distribution, one can now generate a so-called plugin estimate

$$\hat{\theta} = t\left(\hat{F}\right) \tag{3.279}$$

Nonparametric bootstrap provides a modular way of studying the sampling properties (e.g., bias, standard error, quantiles) of the estimate $\hat{\theta}$. Figure 3.39 provides an overview of this procedure.

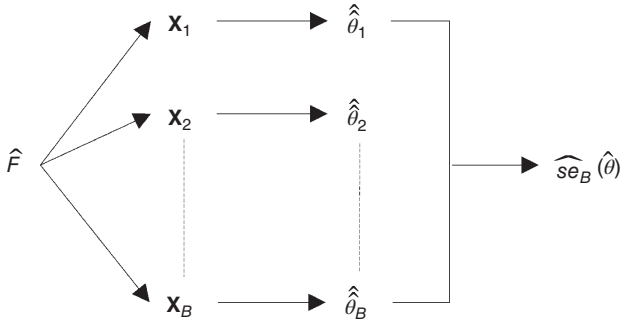


Figure 3.38 Summary of the bootstrap procedure.

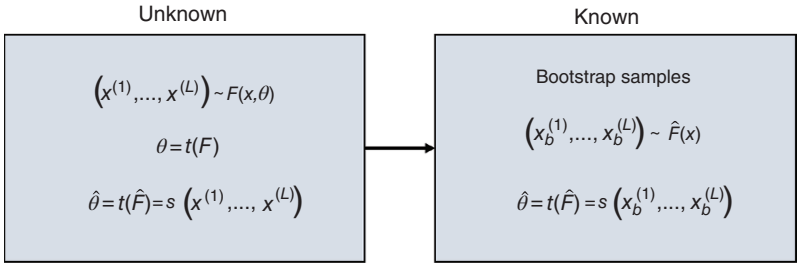


Figure 3.39 The one-sample bootstrap.

3.13.2.2. Bias Correction. A bias is a systematic error made by an estimator about an unknown quantity. Mathematically, this is represented as

$$\begin{aligned} \text{bias}_F &= E_F\left[\hat{\theta}\right] - \theta \\ &= E_F\left[t\left(\hat{F}\right)\right] - t(F) \quad \text{plugin} \end{aligned} \tag{3.280}$$

The bootstrap can be used to estimate this bias as follows:

$$\begin{aligned} \text{bias}_F &= E_F\left[\hat{\theta}\right] - t\left(\hat{F}\right) \\ &= E_{\hat{F}}\left[\hat{\theta}\right] - \hat{\theta} \end{aligned} \tag{3.281}$$

This bias estimate can then be used to correct the original estimate into a bias-corrected estimate:

$$\hat{\theta}_{\text{bias}-\text{corrected}} = 2\hat{\theta} - E_{\hat{F}}\left[\hat{\theta}\right] \tag{3.282}$$

The main issue with bias-corrected estimators is that they may have higher variance. This is clear from Eq. (3.282) since the bias corrections have two quantities that need to be estimated (hence subject to estimation variance). In addition, the multiplication by two increases this effect.

3.13.2.3. Time Series. Bootstrap can also address more complex data structures than the simple one-sample model with single parameter θ. The idea depicted in Figure 3.39 can be extended to any probability model generated the "sample," whether these are time series, maps, 3D cubes of data, and so on. As illustration, consider a time series modeled as a first-order auto-regressive model

$$x(t) = \beta x(t-1) + \varepsilon(t) \tag{3.283}$$

given an observed time series ($x(1)$, …, $x(T)$), β can be estimated as $\hat{\beta}$ through a simple least-square procedure (minimizing the squares error of residuals). In addition, it is assumed that the error follows a certain distribution F. Hence, the generating probability model for the data has two unknowns: (β, F). The empirical distribution of the residuals is

$$\left(\varepsilon(t) = x(t) - \hat{\beta}x(t-1), \frac{1}{T}\right), \quad t = 1, \ldots, T \tag{3.284}$$

which then allows generation of bootstrap samples x_b by the following recursion:

$$x_b(i) = \hat{\beta}x_b(i-1) + \varepsilon_b(i-1) \tag{3.285}$$

3.13.2.4. Regression. Another application of a more complex data structure offers itself in regression. For example, we may want to build a regression between model parameters and data, predictions and data, and so on in the context of UQ (see Chapter 7). Because we have limited samples to do so, we need to get some idea of confidence in that regression. In regression, the aim is to model the conditional expectation on some variable Y given obnservations (independent variables) $\mathbf{X}$, of dimension N. Consider the general situation of samples

$$\mathbf{z}^{(\ell)} = \left(\mathbf{x}^{(\ell)}, y^{(\ell)}\right), \ell = 1, \ldots, L \tag{3.286}$$

The aim is to model the conditional expectation with a linear function

$$E[Y|\mathbf{X}] = \beta^T\mathbf{x} \tag{3.287}$$

The probability structure for the y-values is expressed by means of an error model

$$y = \boldsymbol{\beta}^T\mathbf{x} + \varepsilon \tag{3.288}$$

The classical solution (see Eq. (3.130)) is

$$\hat{\boldsymbol{\beta}} = \left(X^TX\right)^{-1}X^T\mathbf{y} \quad \mathbf{y} = \left(y^{(1)}, \ldots, y^{(L)}\right) \tag{3.289}$$

To apply the bootstrap, we need to first state the probabilistic generating structure $P(\boldsymbol{\beta}, F)$. Here F is the distribution of the residuals ε, which is modeled empirically as $\left(\hat{\boldsymbol{\beta}}, \hat{F}\right)$

$$\hat{F}: P(\varepsilon = \hat{\varepsilon}_\ell) = \frac{1}{L}\hat{\varepsilon}_\ell = y^{(\ell)} - \hat{\boldsymbol{\beta}}^T\mathbf{x}^{(\ell)} \tag{3.290}$$

This allows generating bootstrap samples as follows: estimate $\hat{\boldsymbol{\beta}}$, generate bootstrap sample of ε using Eq. (3.290), and then generate bootstrap samples of y using Eq. (3.288).

3.13.3. Bootstrap with Correlated Data

In geosciences one often deals with models, data, or predictions that are correlated in space and/or time. In such cases, one is equally interested in stating confidence on same statistical parameter whether it is the global mean of a spatial field or parameters of the spatial covariance function, modeling the observations.

A simple approach to deal with such correlated data is to first decorrelate [*Solow*, 1985] them, apply a standard one-sample nonparametric bootstrap, and then back-transform the uncorrelated bootstrap sample to a correlated one. Consider a random vector $\mathbf{X} = (X_1, \ldots, X_N)$ whose elements are correlated (could be space, time, or just multivariate). Then the covariance matrix C_X (modeled from these observations) can be decomposed by means of a Cholesky decomposition:

$$C_X = LL^T \tag{3.291}$$

Decorrelated observations can now be generated as follows:

$$\mathbf{u} = L^{-1}\mathbf{x} \quad C_U = I_N \tag{3.292}$$

Generating any bootstrap sample $\mathbf{u}_b$ results in a boostrap sample $\mathbf{x}_b$

$$\mathbf{x}_b = L\mathbf{u}_b \quad b = 1, \ldots, B \tag{3.293}$$

from which any statistics (e.g., the global mean) can be calculated. Note that this model assumes Gaussianity; hence, some prior normal-score transformation may be required if the X_n are not Gaussian.

Consider now the specific case of a spatial model (a grid with unknown quantities) of which we have some partial observations (e.g., measurements at certain locations). The spatial model may not necessarily be a multi-Gaussian model with some covariance function. For example, the random field may be modeled with a Boolean model, a Markov random field, or some multiple-point geostatistical method (see Chapter 6). The idea of these methods is to generate posterior samples from some (implicit or explicit) posterior model of the unknown grid variables:

$$\left(x(\mathbf{s}_1), \ldots, x\left(\mathbf{s}_{N_{\text{grid}}}\right)\right) \tag{3.294}$$

from some limited set of observations on that grid $\left(x\left(\mathbf{s}_{(1)}\right), \ldots, x\left(\mathbf{s}_{(N_{\text{data}})}\right)\right)$. In a general method for spatial bootstrap, these conditional samples/realizations are resampled using the same sampling strategy (but different locations) to obtain resampled data sets:

$$\left(x_b\left(\mathbf{s}_{(1)}\right), \ldots, x_b\left(\mathbf{s}_{(N_{\text{data}})}\right)\right), b = 1, \ldots, B \tag{3.295}$$

Thus, resampling accounts for whatever correlation structure is assumed in generating the realizations. The bootstrap samples can then be used in any estimator:

$$t\left(x\left(\mathbf{s}_{(1)}\right), \ldots, x\left(\mathbf{s}_{(N_{\text{data}})}\right)\right) \rightarrow t\left(x_b\left(\mathbf{s}_{b,(1)}\right), \ldots, x_b\left(\mathbf{s}_{b,(N_{\text{data}})}\right)\right) \tag{3.296}$$

Consider the case of estimating the global mean of the domain and requiring some confidence on that estimate. The variability of this arithmetic estimator of that unknown mean is dependent on the correlation structure of the field (the pure random case would then be solved with the i.i.d bootstrap, but otherwise this method would yield incorrect confidence). This type of bootstrap allows accounting for that structure. Example applications of these ideas are presented in *Journel* [1994] and *Caumon et al.* [2009].

3.13.4. Bootstrap Confidence Intervals and Hypothesis Testing

In the application of UQ, it is often important to know if two sample sets follow the same distribution or a different distribution. Knowing whether two (empirical) distributions are different is, for example, relevant to the application of Bayes' rule. If the prior is not sufficiently different from the posterior, then clearly the data was not able to reduce uncertainty. Another application lies in sensitivity analysis: we would like to know if a variable/parameter is impacting the response or not. One way of testing this is to classify the response into two groups (positive/negative or high/low or reacting/non-reacting), then study the distribution of that variable in each group (see Chapter 5). If the distribution within the two groups is the same, then the parameter is not impacting the response; likewise, the degree of difference can inform how impacting that variable is with respect to other variables.

The problem is that typically we have a limited sample only. Either because the data is too expensive to acquire, or when computer models are involved, the amount of runs is limited. A hypothesis test is needed where the null hypothesis is defined as "no difference in the distributions exists" and the statistical evidence is used to test if it can be rejected (hypothetico-deductive reasoning, see Chapter 5). Because hypothesis testing requires sampling statistics, bootstrap is an ideal candidate for those tests that involve statistics whose sampling distributions are not known.

Here we present two different ways of addressing these hypotheses tests, each addressed with a different bootstrap method. First consider the null-hypothesis that two distributions are the same:

$$H_0 : F_1 = F_2 \tag{3.297}$$

with $F_1$and F_2 the two distributions in question. Next we define a test statistics, such as for example the difference in mean:

$$\theta = \mu(F_1) - \mu(F_2) \tag{3.298}$$

The difference is also estimated from the data as $\hat{\theta}$. Clearly, if $\hat{\theta}$ is significantly different from zero then the null-hypothesis should be rejected. To study this, we would like to generate bootstrap samples and use them to calculate bootstrap estimates $\hat{\hat{\theta}}_b$, which allows calculating the so-called "achieved significance level" (ASL):

$$\text{ASL} = P_{H_0}\left(\hat{\hat{\theta}}_b \geq \hat{\theta}\right) \tag{3.299}$$

The smaller ASL, the more evidence against the null-hypothesis. The problem now lies in how to resample (a question of the probability generating structure) to generate these bootstrap estimates. Indeed, H_0 leaves many possibilities for distributions $F_1 = F_2$ with a given test statistics (e.g., if the test statistics is the mean only, then many distributions can be constructed). To address this, a permutation test looks at every possible combination of creating two groups with the original sample values. For example, if we have L_1 samples in group 1 (F_1) and

L_2 samples in group 2 (F_2), then the amount of possible combinations is

$$\binom{L_1+L_2}{L_1} = \binom{L_1+L_2}{L_2} = \frac{(L_1+L_2)!}{L_1!L_2!} \tag{3.300}$$

and each of such a combination has probability $1/\binom{L_1+L_2}{L_1}$. A permutation test can be considered as a bootstrap *without* replacement. It selects L_1 samples randomly from the combined set of $L_1 + L_2$ samples, assigns it to group 1 and the remainder to group 2. As a result, samples in group 1 do not replicate in group 2 (that would be a bootstrap *with* replacement). When using a finite amount B of such bootstrap samples, then ASL in Eq. (3.299) can be approximated by

$$\text{ASL}_{perm} = \#\left(\hat{\theta}_b \geq \hat{\theta}\right) / \binom{L_1+L_2}{L_1} \tag{3.301}$$

A second and different way of addressing the same problem is to avoid the original null-hypothesis and directly test

$$H_0 : \theta = \theta_0 \tag{3.302}$$

The distributions are now simply the empirical distributions; hence, B bootstrap samples with replacement are drawn of size $L_1 + L_2$ of which the first L_1 are assigned to group one and the remainder to group 2. Unlike the permutation test, this way of testing does not assign probabilities to each sample (no explicit generating structure). This also means that the ASL obtained $\text{ASL}_{boot} = \#\left(\hat{\theta}_b \geq \hat{\theta}\right)/B$ has no interpretation of a probability as B goes to infinity.

The bootstrap sampling results from hypothesis testing can equally be used to construct confidence intervals on any statistic $\hat{\theta}$. The bootstrap estimates $\hat{\hat{\theta}}_b$ represent the empirical distribution from which any percentiles $\hat{F}^{-1}(\alpha)$ can be calculated. This simply means that for a given α, one finds amongst the B bootstrap samples the sample with rank $(\alpha \times B)/100$. This results in the confidence interval

$$\left[\hat{\theta}_{\alpha,Lo}, \hat{\theta}_{\alpha,Hi}\right] = \left(\hat{\hat{F}}_b(\alpha), \hat{\hat{F}}_b(1-\alpha)\right) \tag{3.303}$$

The link with hypothesis testing also becomes clear now. Consider the case again of testing difference

$$H_0 : \theta = 0 \Rightarrow \text{ASL} = \alpha \tag{3.304}$$

meaning that we can use confidence intervals to calculate ASL. Indeed, consider that $\hat{\theta} > 0$, if we chose α such that $\hat{\theta}_{\alpha,Lo} = 0$ then as a result

$$P_{\theta=0}\left(\hat{\hat{\theta}}_b \geq \hat{\theta}\right) = \alpha \tag{3.305}$$

REFERENCES

Aihara, S. I., A. Bagchi, and S. Saha (2009), On parameter estimation of stochastic volatility models from stock data using particle filter: Application to AEX index, *Int. J. Innov. Comput. I.*, *5*(1), 17–27.

Andrieu, C., A. Doucet, and R. Holenstein (2010), Particle Markov chain Monte Carlo methods, *J. R. Stat. Soc. Ser. B Stat. Methodol.*, *72*, 269–342, doi:10.1111/j.1467-9868.2009.00736.x

Beaumont, M. A., W. Zhang, and D. J. Balding (2002), Approximate Bayesian computation in population genetics, *Genetics*, *162*(4), 2025–2035.

Beven, K. (2009), *Environmental Modelling: An Uncertain Future?* Available from: http://books.google.dk/books/about/Environmental_Modelling.html?id=A_YXIAAACAAJ&pgis=1.

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Clarendon Press, pp. 482, doi:10.2307/2965437.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984), *Classification and Regression Trees*, The Wadsworth Statisticsprobability Series, vol. *19*, doi:10.1371/journal.pone.0015807.

Caers, J. (2011), *Modeling Uncertainty in the Earth Sciences*. Wiley, Hoboken, NJ.

Caumon, G., P. Collon-Drouaillet, C. Le Carlier De Veslud, S. Viseur, and J. Sausse (2009), Surface-based 3D modeling of geological structures, *Math. Geosci.*, *41*(8), 927–945, doi:10.1007/s11004-009-9244-2.

Craiu, R. V., J. Rosenthal, and C. Yang (2009), Learn from thy neighbor: Parallel-chain and regional adaptive MCMC, *J. Am. Stat. Assoc.*, *104*(488), 1454–1466, doi:10.1198/jasa.2009.tm08393.

Cressie, N. A. C. (1985), Fitting variogram models by weighted least squares, *J. Int. Assoc. Math. Geol.*, *17*(5), 563–586, doi:10.1007/BF01032109.

Cressie, N. A. C. (1993), *Statistics for Spatial Data (Revised Edition)*. Wiley, New York, doi:10.2307/2533238.

Davies, D. L., and D. W. Bouldin (1979), A cluster separation measure, *IEEE Trans. Pattern Anal. Mach. Intell.*, *1*(2), 224–227, doi:10.1109/TPAMI.1979.4766909.

Dellaert, F., D. Fox, W. Burgard, and S. Thrun (2001), Monte Carlo localization for mobile robots, *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, February, 1322–1328, doi:10.1109/ROBOT.1999.772544.

Deutsch, C. V., and A. G. Journel (1992), *GSLIB: Geostatistical Software Library and User's Guide*. Oxford University Press, doi:10.1016/0098-3004(94)90041-8.

Diggle, P. J., and P. J. Ribeiro (2007), *Model-Based Geostatistics*, Springer Series in Statistics, vol. *1*, doi:10.1111/1467-9876.00113.

Diggle, P. J., and R. J. Gratton (1984), Monte Carlo methods of inference for implicit statistical models. *J. R. Stat. Soc. Ser. B Methodol.*, *46*, 193–227.

Dubuisson, M. P., and A. K. Jain (1994), A modified Hausdorff distance for object matching, *Proceedings of 12th International Conference on Pattern Recognition*, Jerusalem, Israel, vol. 1, 566–568, doi:10.1109/ICPR.1994.576361.

Efron, B. (1979), Bootstrap methods: Another look at the jackknife, *Ann. Stat.*, *7*(1), 1–26, doi:10.1214/aos/1176344552.

Efron, B., and R. J. Tibshirani (1994), *An Introduction to the Bootstrap*, CRC Press, Boca Raton, FL.

Feyen, L., and J. Caers (2006), Quantifying geological uncertainty for flow and transport modeling in multi-modal heterogeneous formations, *Adv. Water Resour.*, *29*(6), 912–929, doi:10.1016/j.advwatres.2005.08.002.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2004), *Bayesian Data Analysis*, Chapman Texts in Statistical Science Series, doi:10.1007/s13398-014-0173-7.2.

Geman, S., and D. Geman (1984), Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Mach. Intell.*, *6*(6), 721–741, doi:10.1109/TPAMI.1984.4767596.

Geyer, C. J. (2002), *Introduction to Markov Chain Monte Carlo*, no. 1990, pp. 3–48.

Gilks, W. R., G. O. Roberts, and E. I. George (1994), Adaptive direction sampling, *J. R. Stat. Soc., Ser. D*, *43*(1), 179–189, doi:10.2307/2348942.

Gómez-Hernández, J. J, and X.-H. Wen (1998), To be or not to be multi-Gaussian? A reflection on stochastic hydrogeology, *Adv. Water Resour.*, *21*(1), 47–61, doi:10.1016/S0309-1708(96)00031-0.

Goovaerts, P. (1997), *Geostatistics for Natural Resources Evaluation*. Oxford University Press, New York.

Gupta, H. V., M. P. Clark, J. A Vrugt, G. Abramowitz, and M. Ye (2012), Towards a comprehensive assessment of model structural adequacy, *Water Resour. Res.*, doi:10.1029/2011WR011044.

Haario, H., E. Saksman, and J. Tamminen (1999), Adaptive proposal distribution for random walk metropolis algorithm, *Comput. Stat.*, *14*(3), 375, doi:10.1007/s001800050022.

Haario, H., E. Saksman, and J. Tamminen (2001), An adaptive metropolis algorithm, *Bernoulli*, *7*(2), 223, doi:10.2307/3318737.

Hansen, T. M., A. G. Journel, A. Tarantola, and K. Mosegaard (2006), Linear inverse Gaussian theory and geostatistics, *Geophysics*, *71*(6), R101, doi:10.1190/1.2345195.

Hansen, T. M., K. S. Cordua, and K. Mosegaard (2012), Inverse problems with non-trivial priors: Efficient solution through sequential Gibbs sampling, *Comput. Geosci.*, *16*(3), 593–611, doi:10.1007/s10596-011-9271-1.

Hastie, T., R. Tibshirani, and J. Friedman (2009), *The Elements of Statistical Learning Data Mining, Inference, and Prediction*, Second Edition, Springer Series in Statistics, doi:10.1007/978-0-387-84858-7.

Hastings, W. K. (1970), Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, *57*(1), 97–109, doi:10.1093/biomet/57.1.97.

Huttenlocher, D. P., G. A. Klanderman, and W. J. Rucklidge (1993), Comparing images using the Hausdorff distance, *IEEE Transactions on pattern analysis and machine intelligence*, *15*(9), 850–863.

Jain, A. K. (2010), Data clustering: 50 years beyond K-means, *Pattern Recogn. Lett.*, *31*(8), 651–66, doi:10.1016/j.patrec.2009.09.011.

Jaynes, E. T. (2003), Probability theory: The logic of science, *Math. Intell.*, *27*(2), 83–83, doi:10.1007/BF02985800.

Journel, A. G. (1994), Resampling from stochastic simulations, *Environ. Ecol. Stat.*, *1*(1), 63–91, doi:10.1007/BF00714200.

Journel, A. G., and C. V. Deutsch (1993), Entropy and spatial disorder, *Math. Geol.*, *25*(3), 329–355, doi:10.1007/BF00901422.

Journel, A. G., and C. J. Huijbregts (1978), *Mining Geostatistics*, Academic Press, London.

Kaufman, L., and P. J. Rousseeuw (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York.

Kwok, J. T. Y., and I. W. H. Tsang (2004), The pre-image problem in Kernel methods, *IEEE Trans. Neural Netw.*, *15*(6), 1517–1525, doi:10.1109/TNN.2004.837781.

Laloy, E., and J. A. Vrugt (2012), High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing, *Water Resour. Res.*, *48*(1), W01526, doi:10.1029/2011WR010608.

Leeuwen, V., and P. Jan (2009), Particle filtering in geophysical systems, *Mon. Weather Rev. 137*, 4089–4114, doi:10.1175/2009MWR2835.1.

Lemieux, C. (2009). *Monte Carlo and Quasi-Monte Carlo Sampling*, Springer Science & Business Media, New York.

Liu, J. S., F. Liang, and W. H. Wong, (2000), The multiple-try method and local optimization in metropolis sampling, *J. Am. Stat. Assoc.*, *95*(449), 121–134.

Mariethoz, G., and J. Caers (2014), *Multiple-Point Geostatistics: Stochastic Modeling with Training Images*, Wiley Blackwell, Chichester.

Matheron, G. (1970), *La Théorie Des Variables Régionalisées*. Les Cahiers du Centre de morphologie mathématique de Fontainebleau, École Nationale Supérieure des Mines.

Mckay, D. J. C. (1998), Introduction to Monte Carlo methods, in *Learning in Graphical Models*, edited by M. I. Jordan, pp. 175–204, Kluwer Academic Press.

Mosegaard, K., and A. Tarantola (1995), Monte Carlo sampling of solutions to inverse problems, *J. Geophys. Res.*, *100*(B7), 12431–12447, doi:10.1029/94JB03097.

von Mises, R. (1964), *Mathematical Theory of Probability and Statistics*, Academic Press, New York.

Nummiaro, K., E. Koller-Meier, and L. Van Gool (2003), An adaptive color-based particle filter, *Image Vis. Comput. 21*(1), 99–110, doi:10.1016/S0262-8856(02)00129-4.

Ramsay, J., and B. W. Silverman (2005), *Functional Data Analysis*, Springer Series in Statistics, Springer, New York.

Sadegh, M., and J. A. Vrugt (2013), Bridging the gap between GLUE and formal statistical approaches: approximate Bayesian computation, *Hydrol. Earth Syst. Sci.*, *17*(12), 4831–4850, doi:10.5194/hess-17-4831-2013.

Sadegh, M., and J. A. Vrugt (2014), Approximate Bayesian computation using Markov chain Monte Carlo simulation, *Water Resour. Res.*, *10*(2), 6767–6787, doi:10.1002/2014WR015386.Received.

Scheidt, C., and J. Caers (2009), Representing spatial uncertainty using distances and Kernels, *Math. Geosci.*, *41*(4), 397–419, doi:10.1007/s11004-008-9186-0.

Scheidt, C., and J. Caers (2013), Uncertainty quantification in reservoir performance using distances and kernel methods: Application to a west Africa deepwater turbidite reservoir, *SPE J.*, *14*(4), 680–692, doi:10.2118/118740-PA.

Schölkopf, B., and A. J. Smola (2002), *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge.

Schoups, G., and J. A. Vrugt (2010), A formal likelihood function for parameter and predictive inference of hydrologic models with correlated, heteroscedastic, and non-Gaussian errors, *Water Resour. Res.*, *46*, 1–17, doi:10.1029/2009WR008933.

Silverman, B. W. (1986), Density estimation for statistics and data analysis, *Chapman and Hall*, *37*(1), 1–22, doi:10.2307/2347507.

Solow, A. R. (1985), Bootstrapping correlated data, *J. Int. Assoc. Math. Geol.*, *17*(7), 769–775, doi:10.1007/BF01031616.

Storn, R., and K. Price (1997), Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.*, *11*(4), 341–359, doi:10.1023/A:1008202821328.

Tarantola, A. (1987). *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*, Elsevier, Amsterdam. Available from: http://www.scopus.com/inward/record.url?eid=2-s2.0-0023499373&partnerID=tZOtx3y1.

Ter Braak, C. J. F. (2006), A Markov chain Monte Carlo version of the genetic algorithm differential evolution: Easy Bayesian computing for real parameter spaces, *Stat. Comput. 16*(3), 239–249, doi:10.1007/s11222-006-8769-1.

Tjelmeland, H., and J. Besag (1998), Markov random fields with higher-order interactions, *Scand. J. Stat.*, *25*, 415–433, doi:10.1111/1467-9469.00113.

Turner, B. M., and T. Van Zandt (2012), A tutorial on approximate Bayesian computation, *J. Math. Psychol.*, *56*(2), 69–85, doi:10.1016/j.jmp.2012.02.005.

Vapnik, V., and A. Lerner (1963), Pattern recognition using generalized portrait method, *Autom. Remote Control*, *24*, 774–780.

Von Neumann, J. (1951), Various techniques used in connection with random digits, Applied Math Series, Notes by G. E. Forsythe, National Bureau of Standards, 12, 36–38.

Vrugt, J. A. (2016), Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB implementation, *Environ. Model. Softw.*, *75*, 273–316, doi:10.1016/j.envsoft.2015.08.013.

Vrugt, J. A, C. J. F. ter Braak, C. G. H. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon (2009), Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling, *Int. J. Nonlin. Sci. Num. Simul.*, *10*(3), 273–290, doi:10.1515/IJNSNS.2009.10.3.273.

Williams, C. (1999), Prediction with Gaussian processes, in *Learning in Graphical Models*, edited by M. I. Jordan, MIT Press, Cambridge, MA, pp. 599–621.

Zinn, B., and C. F. Harvey (2003), When good statistical models of aquifer heterogeneity go bad: a comparison of flow, dispersion, and mass transfer in connected and multivariate Gaussian hydraulic conductivity fields, *Water Resour. Res.*, *39*(3), 1–19, doi:10.1029/2001WR001146.