15 Empirical Orthogonal Function (EOF) Analysis

15.1 Overview

Empirical Orthogonal Function (EOF) analysis is designed to find covariability within a data set and create new composite variables that capture that internal dependence, allowing a few composite uncorrelated variables to describe most of the variability (variance) in the data described in the much larger, dependent dataset. This is typically applied to large space-time data (e.g., time series collected at numerous spatial locations), though it can also be used on individual time series, or even nonsequential data – for example, if you have collected air temperature data every day for 10 years at every airport in the United States.[1] Typically, the different time series will show some degree of dependence (e.g., if it is exceptionally hot in one location, it is likely to be hot in others and just average or even cold in others). This means that different time series in different locations are carrying replicate information. EOF analysis finds those relationships in space that share the same time variability and combines them into a single spatial pattern sharing a common time variability. This pattern is found by regression across space so that a precise relationship between neighboring sites is found. Ultimately, the patterns that are found show these covarying locations, producing, for example, a single pattern in space that shows how any one location covaries with all others. So if it is hot in one location, the pattern shows those locations that are also hot, slightly cooler, a lot cooler, average, etc. This single pattern then captures all of the replicate information otherwise stored in the many individual time series. This makes it easier to view how the variable varies in space and time and collapses a potentially huge data set into a minimum number of patterns that capture most of the variance.

15.2 Introduction

Previously, we have fit data with known functions, such as sines, cosines or polynomials, in an effort to determine the fundamental characteristics of the processes they represent

[1] I particularly like the Philadelphia airport (PHL), where there are many excellent restaurants to eat at during weather-related delays at flight destinations.

or to identify some underlying signal buried within noise. But the signal within your data may reflect the linear combination of a set of processes that do not follow some standard (or any) functional form. For example, the distribution of rainfall within a region may represent a linear combination of the temperature, humidity, atmospheric pressure and other physical variables. Each of these variables changes in time in a highly irregular fashion, defying a simple mathematical description. In such a case, how can you decompose the rainfall data into a set of meaningful functions?

Empirical orthogonal function (**EOF**) analysis, or **eigenvector analysis**, is the tool you employ when searching for a set of natural, or *empirical*, orthogonal functions that can be combined linearly to describe the data.[2] If the data are thought to represent the sum of different periodic functions, then we decompose the data set into a set of orthogonal sinusoids (Fourier harmonics). With EOFs, we are decomposing the data into a set of empirical orthogonal functions – functions with no standard mathematical representation, but instead, shapes that, while orthogonal, represent the regularly occurring, temporally coherent major-variance-describing patterns present in the data (just like sines and cosines are regular patterns that have simple mathematical representation, here we will find shapes that regularly occur, regardless of their particular form). This ability to isolate consistent patterns within individual data sets makes EOFs "*data adaptive*."

EOF analysis finds patterns that share the same temporal variability to be consistent throughout the datasets, so that one pattern, orthogonal to the other patterns, can now be constructed, replacing that same pattern hidden in each time series and being repeated multiple times in the data set. Once this first pattern is identified, another pattern is identified, capturing the greatest amount of the remaining temporal variability not already captured by the previous pattern(s).[3] While this is a statistical construct, we hope that the mechanisms driving the phenomenon being examined may have a unique signature that is captured by this methodology – here identified as a pattern that varies in time the same way in each time series in the dataset. Each of the patterns identified has an associated time series showing how the amplitude of the pattern changes in time, known as **principal components** (**PCs**, or, **expansion coefficients**). Each pattern (the **EOF**) and its time series (PC) together are called a **mode**. Individual modes can be isolated and studied as potentially physically meaningful (though they needn't be so), and a subset of modes can be combined to describe the signal (with the remaining, unused set describing noise).

Consider the following example of a spatio-temporal dataset consisting of a measure of upper-ocean freshwater content (called salt deficit, SD_W), measured within a grid in the waters lying along the western (Pacific) margin of the Antarctic Peninsula (Figure 15.1). This grid was sampled every austral summer for 12 years (now over 20

[2] You can decompose a data set into nonorthogonal functions, but the advantage of orthogonal functions is that they are completely independent of one another, thus any part of the signal described by one orthogonal function is no longer available for description by another. Because of this, the different orthogonal functions can be added together linearly to describe the dataset. And it is because of the orthogonality that the solution for the function coefficients is so incredibly simple (i.e., the Fourier transform, an approach that can be applied to any orthogonal function, not just sinusoids). Orthogonality does introduce some limitations to be relaxed when introducing rotated EOFs (not discussed within).

[3] Formally, we identify the common time series describing the pattern, and then other time series that is orthogonal to the other time series, with each having its own different pattern.

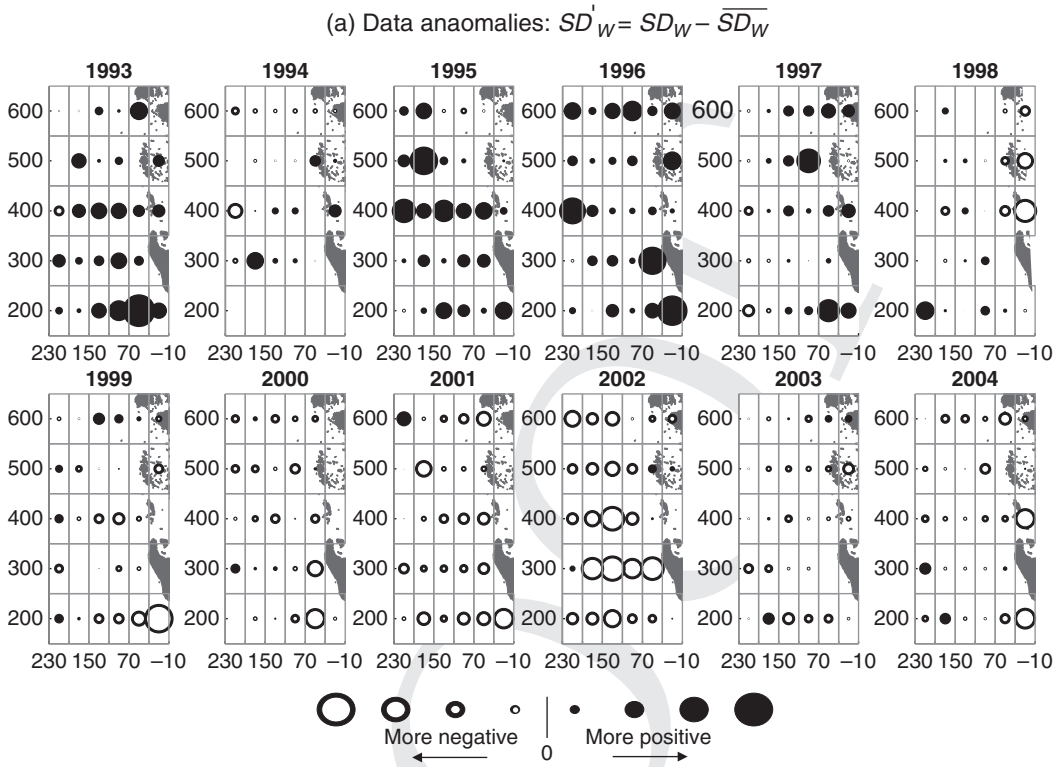


Figure 15.1 (**A)** Anomaly plots (data value minus 12-year average at each grid point). Each year, all anomaly maps share a pattern that has an identical time history. The dominant time history is shown by this first PC (PC1) (**B**), and the pattern undergoing this temporal change (the EOF1) in (**C**). In this case, this first PC describes ~51 percent of the temporal variance, leaving another 49 percent to be described by other patterns. (**D)** Each year, EOF1 is multiplied by PC1 for that year to give the pattern with actual mode 1 values for that year. This primary mode shows a strong decreasing trend: just before 1998, the SD'_W becomes mostly negative (water is saltier), so the sign of this first PC changes from positive to negative to accommodate this (given that all but one cell in the corresponding EOF1 are of the same sign), reaching maximum negative (salty) in 2002 as seen in the anomaly maps, the reconstruction and the PC.

years), giving a *map* of values for each of the 12 years and a 12-year time series in each grid cell. We wish to determine if there is any consistent variability in space and/or time over these 12 years that might help us to better isolate any change leading to an understanding of any physical mechanisms driving such change. For this, we employ empirical orthogonal function analysis.

EOF analysis, as employed within, is a general term that encompasses a family of techniques, such as **principal components analysis** (**PCA**, the statistical name of what I am more generally calling EOF analysis), **canonical correlation analysis** (**CCA**) and **factor analysis**, all of which employ some subtly different form by which the empirical functions are arrived at or treated. Typically, EOF analysis is an exercise in multivariate statistics, and thus it is concerned with decomposing multiple variables into a set of orthogonal functions (equivalent to a multi-dimensional Fourier decomposition).

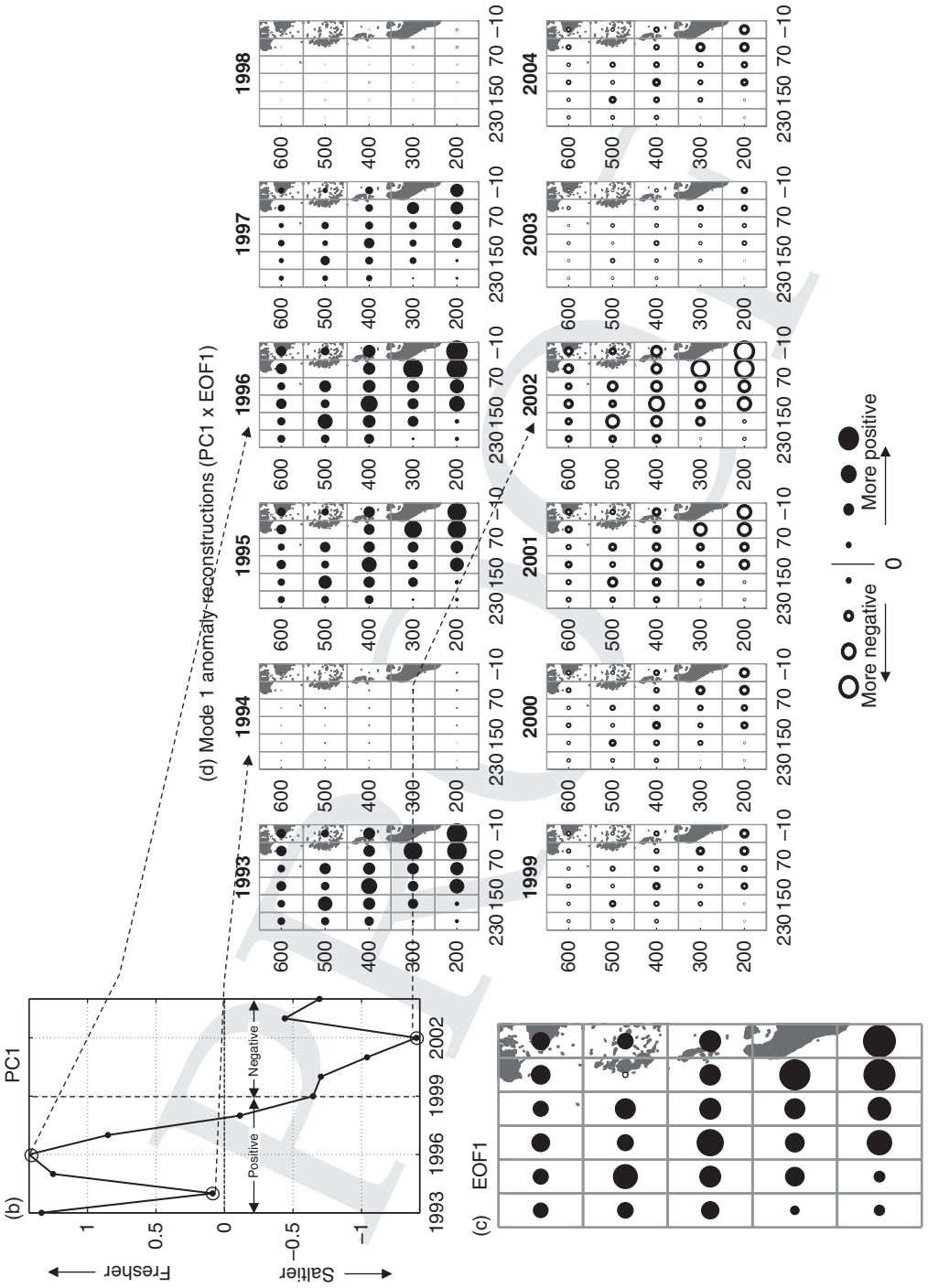


Figure 15.1 (cont.)

The decomposition is invaluable for, among other things, providing the most amount of information in the most manageable form. Consider a situation where we have measured p different variables. The simplest statistics, i.e., the mean, variance and covariance between variables, requires $p + (p^2 + p)/2$ values to describe them. If the variables were uncorrelated, then the covariance is zero between all pairs, and thus the data description would be accomplished with only two p statistics – a considerable reduction, given a large p! EOF techniques are useful for reducing this massive data set to a smaller number of independent, uncorrelated variables (the covariances are zero), thus reducing the *dimension* or *degrees of freedom* of the data set from p to a smaller size, say, k, that contains most of the variance. With these, the data can be described by two k statistics.[4] The techniques are also useful for identifying coherent, even if functionally irregular, structure in the data.

There is a tremendous body of literature associated with these techniques. EOF analysis can also be used to decompose a single time series into a sum of orthogonal functions[5], though traditionally it is more typically used to decompose multivariate or multidimensional data.

15.3 Eigenvector Analysis

15.3.1 Fundamentals

The heart of EOF analysis centers on the concept of **eigenvectors**, also referred to as **characteristic vectors**. Eigenvectors can be described and derived from a variety of perspectives, but the fundamental point here is a property of a *square* matrix **A** that has no counterpart in ordinary algebra.

We begin by asking the question whether it is possible to multiply a vector, **x**, by matrix **A** and produce a new vector, **b**, that is parallel to the original **x**. Or, alternatively, for any square matrix **A**, does there exist one or more non-trivial (i.e., non-zero) vectors **x**, that when multiplied (transformed) by matrix **A**, yield the original, though scaled, vector **x?** In this case,

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{15.1a}$$

or, rearranged,

$$(\mathbf{A} - \boldsymbol{\lambda}\mathbf{I})\mathbf{x} = 0. \tag{15.1b}$$

Here, λ is a scalar such that the new vector $\mathbf{b}$ ($= \lambda\mathbf{x}$) is parallel to **x**, since parallel vectors can differ only in length (or magnitude), and thus are equal to scalar multiples of one another (see Appendix A1 on Matrix Algebra).

Conceptually, one might imagine that satisfying (15.1) should be a rather difficult task, since the operation of **Ax** is equivalent to multiplying every element of **x** by an

[4] Often we will have removed the mean from the original variables for simplicity, in which case the reduced set has zero mean.

[5] Discussed at the end of this chapter, after the fundamentals of the technique have been derived and examined.

entire column in **A**, and yet the net result must be the original vector **x**, altered only by a constant. So,

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ & \vdots & & \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{15.1c}$$

$$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{15.1d}$$

or, expanding the multiplication in terms of the $\mathbf{a}_j$ vectors,

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{bmatrix} + \dots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn} \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{15.1e}$$

Therefore, after a rather elaborate multiplication of each element in **x**, the new vector still points in the same original direction – all of the multiplications doing nothing more than ultimately scaling each element of **x** by the same amount.

Alternatively, the difficulty is clearly apparent by inspection of the system that (15.1) represents,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{13}x_3 &= \lambda x_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= \lambda x_2 \\ \vdots & \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= \lambda x_n, \end{aligned} \tag{15.2a}$$

which tends to emphasize the row vector manipulation taking place in (15.1). That is, the elements of **x** are being multiplied by an entire row of **A**, and yet the final product is still just the scaled **x**. In other words, each element x_i is a weighted sum of all other elements of **x**, yet the weighted sum is simply proportional to the original element. We wish to find those elements of **x** that satisfy this criterion for the "weights" contained in any square matrix **A**.

Equation (15.2a) is rearranged to give

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{13}x_3 - \lambda x_1 &= 0 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n - \lambda x_2 &= 0 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n - \lambda x_n &= 0 \end{aligned} \tag{15.2b}$$

or

$$\begin{aligned}(a_{11} - \lambda)x_1 + a_{12}x_2 + \ldots + a_{13}x_3 &= 0\\ a_{21}x_1 + (a_{22} - \lambda)x_2 + \ldots + a_{2n}x_n &= 0\\ &\vdots\\ a_{n1}x_1 + a_{n2}x_2 + \ldots + (a_{nn} - \lambda)x_n &= 0.\end{aligned} \tag{15.2c}$$

In matrix form,

$$x_1\begin{bmatrix} a_{11}-\lambda \\ a_{21} \\ \vdots \\ a_{n1}\end{bmatrix} + x_2\begin{bmatrix} a_{12} \\ a_{22}-\lambda \\ \vdots \\ a_{n2}\end{bmatrix} + \ldots + x_n\begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn}-\lambda\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0\end{bmatrix} \tag{15.2d}$$

$$\begin{bmatrix} a_{11}-\lambda & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22}-\lambda & & a_{2n} \\ & \vdots & & \\ a_{n1} & a_{n2} & & a_{nn}-\lambda\end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n\end{bmatrix} = 0. \tag{15.2e}$$

Thus, given carefully chosen elements of **x** and the constant λ, it is possible to sum up the column vectors in the (**A** − λ**I**) matrix so that they sum to the **null vector** – that is, they cancel one another out. The vectors of (**A** − λ**I**), when scaled by the elements of **x**, ultimately end up back at the origin.[6]

In this *homogeneous* form (i.e., the equation equals zero), we know that the system (15.2) can only have a nontrivial solution when the determinant is equal to zero:

$$\begin{vmatrix} a_{11}-\lambda & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22}-\lambda & & a_{2n} \\ & \vdots & & \\ a_{n1} & a_{n2} & & a_{nn}-\lambda\end{vmatrix} = 0. \tag{15.3}$$

That is, the system shown in (15.2e), or represented by the equivalent matrix expression of (15.1b), shows that the weighted sum of the columns in matrix (**A** − λ**I**) sum to zero, and this is equivalent to the definition of *dependent* columns. That is, the columns of a matrix are dependent if they can be summed to zero when weighted with the appropriate weights. A matrix with dependent columns is singular (see Appendix 1), and a singular matrix has its determinant equal to zero – thus (15.3) above. (**A** − λ**I**) must be singular if a solution to (15.1) or (15.2) exists, other than the trivial solution (i.e., **x** = **0**), which is rather meaningless.

The determinant in (15.3) can be expanded to form a polynomial in λ, normalized by multiplying through by (−1)n for convenience,[7] as

[6] The addition of vectors can be done graphically by drawing each vector with a line from its origin to its tip, where its length is proportional to its magnitude and its angle reflects its direction. Then, each vector is added to the previous one by simply placing its origin on the tip of the previous vector. The sum is then given by the vector drawn from the origin (the location of the origin of the first vector in the sum) to the tip of the final vector – in this case, the final vector tip ends at the origin (i.e., we have just undone all of the additions to get back to where we started).

[7] This normalization, or simple scaling, is done to ensure that the polynomial is always written in the form of (15.4). If not done, for *n* equal to an odd number, the first term in the sum, λ^n, would be $-\lambda^n$.

$$\lambda^n + c_{n-1}\lambda^{n-1} + c_{n-2}\lambda^{n-2} + \ldots + c_1\lambda^1 + c_0 = 0. \tag{15.4}$$

This polynomial in λ is called the characteristic equation, characteristic function or characteristic polynomial.

For example, if n = 2, then, the determinant in (15.3) is expanded as

$$\begin{aligned} |\mathbf{A} - \lambda\mathbf{I}| &= (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} \\ &= \lambda^2 - a_{11}\lambda - a_{22}\lambda + a_{11}a_{22} - a_{12}a_{21} \\ &= \lambda^2 + (-a_{11} - a_{22})\lambda + a_{11}a_{22} - a_{12}a_{21} \\ &= \lambda^2 + c_1\lambda + c_0 = 0. \end{aligned} \tag{15.5}$$

This characteristic equation is solved by the quadratic formula,

$$\lambda = \frac{-c_1 \pm \sqrt{c_1^2 - 4c_0}}{2}, \tag{15.6a}$$

or by factoring and solving for the *roots*:

$$(\lambda + b_1)(\lambda + b_2) = 0. \tag{15.6b}$$

The equation has two solutions: one, λ_1, when taking the "+" in the numerator of (15.6a); the other, λ_2, when taking the "–". These correspond to the two roots, $\lambda_1 = -b_1$ and $\lambda_2 = -b_2$ in (15.6b). So there are two different values of λ that satisfy this determinant and satisfy $\mathbf{Ax} = \lambda\mathbf{x}$.

Generalized, a polynomial of order *n*, and thus the determinant of (15.3), will always have *n* (and only *n*) roots that satisfy the equation (though all *n* roots needn't be unique).[8] These *n* roots, that is, the *n* values of λ that satisfy (15.1) are called the **eigenvalues**[9] of the matrix **A**.

For each eigenvalue, $\lambda_1, \lambda_2, \ldots, \lambda_n$, there exists an explicit system (15.2c) that is satisfied. So, for the case of $n = 2$, there exist

$$\begin{aligned} (a_{11} - \lambda_1)x_1 + a_{12}x_2 &= 0 \\ a_{21}x_1 + (a_{22} - \lambda_1)x_2 &= 0 \end{aligned} \tag{15.7a}$$

and

$$\begin{aligned} (a_{11} - \lambda_2)x_1 + a_{12}x_2 &= 0 \\ a_{21}x_1 + (a_{22} - \lambda_2)x_2 &= 0. \end{aligned} \tag{15.7b}$$

Given the values of λ_1 and λ_2, these two systems of equations (15.7a) and (15.7b) can be solved for the two **x** vectors, which are those that are simply scaled when multiplying the matrix **A** for which the eigenvalues correspond. These vectors that satisfy (15.1) and which exist for each eigenvalue, are called **eigenvectors**.

[8] The solution of an *n*th order polynomial for the roots is not a trivial task, and is best obtained using one of several numerical techniques, such as singular value decomposition (SVD), for example.

[9] Other names often associated with these are "characteristic values," "proper roots," "proper values," "λ-roots" or "latent roots." These names "proper," "latent" and "characteristic" also can be used in place of "eigen" in the previously defined terms.

So, in (15.7a), the coefficients of the system, $(a_{11} - \lambda_1)$ and $(a_{22} - \lambda_1)$, yield values of x_{11} and x_{21}, forming the first eigenvector $\mathbf{x}_1$, that are different from the values of x_{12} and x_{22}, forming the second eigenvector, $\mathbf{x}_2$, resulting from $(a_{11} - \lambda_2)$ and $(a_{22} - \lambda_2)$.[10]

In the general case, with n eigenvalues, each eigenvalue, obtained by solving the determinant in (15.3), leads to an eigenvector, **e**, obtained by solving the system (15.2) for that eigenvalue:

$$\begin{aligned} \lambda_1 \text{ yields } \mathbf{e}_1 &= [x_{11}\ x_{21}\ x_{31}\ \ldots\ x_{n1}]^{\mathrm{T}} \\ \lambda_2 \text{ yields } \mathbf{e}_2 &= [x_{12}\ x_{22}\ x_{32}\ \ldots\ x_{n2}]^{\mathrm{T}} \\ &\vdots \\ \lambda_n \text{ yields } \mathbf{e}_n &= [x_{1n}\ x_{2n}\ x_{3n}\ \ldots\ x_{nn}]^{\mathrm{T}}. \end{aligned} \tag{15.8}$$

Each eigenvalue, λ_j, allows the equation $\mathbf{Ax} = \lambda\mathbf{x}$ to be solved by the corresponding eigenvector, $\mathbf{x}_j$. Note that, since these eigenvalue-eigenvector pairs represent solutions to homogeneous equations, we can multiply any of the eigenvectors by a constant and still satisfy the equations. Therefore, this is consistent with the original premise that we are finding vectors that, when multiplied by a matrix, retain their original direction, but not necessarily their original length (magnitude). Thus, it is the *direction* of the eigenvectors, not their magnitude, that is unique for each eigenvector. Typically, we work with the eigenvectors after normalizing them to unit length: $\mathbf{e} = \mathbf{e}/(\mathbf{e}^{\mathrm{T}}\mathbf{e})^{1/2}$.

Once we have all of the eigenvalues and corresponding eigenvectors, we can represent the complete **eigenstructure** – that is, the entire set of eigenvalues and eigenvectors – in a single equation of the form

$$\mathbf{AE} = \mathbf{E\Lambda}, \tag{15.9a}$$

where **A** is the original square matrix from which the eigenvalues and eigenvectors originate, **E** is the square $(n \times n)$ matrix for which each column contains one eigenvector, and $\boldsymbol{\Lambda}$ (sometimes denoted as **L**) is a square diagonal matrix $(n \times n)$ that contains an eigenvalue on each diagonal element, such that the eigenvalues are aligned with their corresponding eigenvector in the **E** matrix[11], as

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{pmatrix} \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_1 e_1 & \lambda_2 e_2 & \cdots & \lambda_n e_n \end{bmatrix} \tag{15.9b}$$

[10] Actually, the two eigenvalues can be the same, but the point is that there will be a distinct eigenvector for each eigenvalue, regardless of whether they are the same or not.

[11] Post-multiplying by a diagonal matrix scales the columns of a matrix, whereas pre-multiplying scales the rows.

Box D15.1 Natural Eigenfunctions of Linear Systems

Consider the complex function $e^{i\omega t}$ in the context of linear systems, where a linear system has the property

$$L(a\mathbf{x} + b) = aL(\mathbf{x}) + b \tag{D15.1.1}$$

and L is a linear operator such as differentiation, integration, convolution, etc.

As an example, consider convolution:

$$h_t = \sum_{j=0}^{N-1} f_j g_{t-j}. \tag{D15.1.2a}$$

Here, f is the time-invariant impulse response function of a filter, h is the output from the filter and g is the input, both of which are functions of time (or lag), indexed by t. If g is given by the complex exponential, we have

$$\begin{aligned} h_t &= \sum_{j=0}^{N-1} f_j e^{i\omega(t-j)} \\ &= \sum_{j=0}^{N-1} f_j e^{i\omega t} e^{-i\omega j} \\ &= e^{i\omega t} \sum_{j=0}^{N-1} f_j e^{-i\omega j} \\ &= c(\omega)\, e^{i\omega t}, \end{aligned} \tag{D15.1.2b}$$

where

$$c(\omega) = \sum_{j=0}^{N-1} f_j e^{-i\omega j}, \tag{D15.1.2c}$$

which is the Fourier transform of the impulse response function, the **transfer function** of the filter (Chapter 13).

Therefore, the input function $e^{i\omega t}$, when operated upon by a linear filter, gives an output function that is proportional to the input function. In other words, *the complex exponential is a natural eigenfunction for this (and all) linear systems*. The function of eigenvalues, c(ω), is nothing more than the transfer function for the system.

Because we know that the complex exponential is the complex sum of a sine and cosine (as stated by Euler's law), this suggests that the sines and cosines used in Fourier analysis to fit time series are the natural eigenvectors to linear systems. This is one reason why they are excellent functions to use for fitting and decomposing time series (though they have some other generally beneficial properties as well).

15.3.2 Orthogonality for Symmetrical Matrices

Now consider a symmetrical matrix **A** and any two of its eigenvalues and corresponding (normalized) eigenvectors, say λ_i, $\mathbf{e}_i$ and λ_j, $\mathbf{e}_j$, such that

$$\mathbf{A}\mathbf{e}_i = \boldsymbol{\lambda}_i\mathbf{e}_i \tag{15.10a}$$

$$\mathbf{A}\mathbf{e}_j = \boldsymbol{\lambda}_j\mathbf{e}_j. \tag{15.10b}$$

Since **A** is symmetric, **A** = **A**T, and thus the transpose of one of the above equations is

$$(\mathbf{A}\mathbf{e}_i)^{\mathrm{T}} = \boldsymbol{\lambda}_i\mathbf{e}_i^{\mathrm{T}}, \tag{15.11a}$$

or (recalling the reversal rule of transposed products)

$$\mathbf{e}_i^{\mathrm{T}}\mathbf{A} = \boldsymbol{\lambda}_i\mathbf{e}_i^{\mathrm{T}}. \tag{15.11b}$$

Now consider pre-multiplying (15.10b) by **eT** and post-multiplying (15.11b) by $\mathbf{e}_j$ to give

$$\mathbf{e}_i^{\mathrm{T}}\mathbf{A}\mathbf{e}_j = \boldsymbol{\lambda}_j\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_j \tag{15.12a}$$

$$\mathbf{e}_i^{\mathrm{T}}\mathbf{A}\mathbf{e}_j = \boldsymbol{\lambda}_i\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_j. \tag{15.12b}$$

The vector-products of both equations are the same (so the only differences are the eigenvalues), and (15.12) must hold for all i and j, including those (usually the majority) for which $\lambda_i \neq \lambda_j$, therefore the two equations (15.12) can only hold if

$$\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_j = 0. \tag{15.13}$$

The condition that $\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_j = 0$ *shows that (i.e., is the definition that) the eigenvectors, e, are orthogonal to one another*. Furthermore, we have already defined e to be unit length,

$$\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_i = 1, \tag{15.14}$$

so the vectors as defined are in fact, **orthonormal**.

Empirical Orthogonal Functions (EOFs)

These orthogonal eigenvectors (*which result from all symmetric matrices A*), are often termed **empirical orthogonal functions** (**EOFs**). Combining them into a single matrix **E** as in (15.9a) gives an orthonormal matrix. All non-diagonal elements in $\mathbf{E}^{\mathrm{T}}\mathbf{E}$ are the covariances between the vectors (zero for these orthogonal vectors) and the diagonals are the magnitude of the vectors, which were already normalized to equal one. So, $\mathbf{E}^{-1} = \mathbf{E}^{\mathrm{T}}$, $\mathbf{E}^{\mathrm{T}}\mathbf{E} = \mathbf{I} = \mathbf{E}\mathbf{E}^{\mathrm{T}}$, and thus,

$$\begin{aligned} \mathbf{AE} &= \mathbf{E}\Lambda \\ \mathbf{AEE}^{\mathrm{T}} &= \mathbf{E}\Lambda\mathbf{E}^{\mathrm{T}} \\ \mathbf{A} &= \mathbf{E}\Lambda\mathbf{E}^{\mathrm{T}} \end{aligned} \tag{15.15a}$$

or

$$\begin{aligned} \mathbf{AE} &= \mathbf{E}\Lambda \\ \mathbf{E}^{\mathrm{T}}\mathbf{AE} &= \mathbf{E}^{\mathrm{T}}\mathbf{E}\Lambda \\ \mathbf{E}^{\mathrm{T}}\mathbf{AE} &= \Lambda, \end{aligned} \tag{15.15b}$$

where

$$\mathbf{E}^{\mathrm{T}}\mathbf{E} = \mathbf{E}\mathbf{E}^{\mathrm{T}} = \mathbf{I} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \\ & & & 1 \end{bmatrix}. \tag{15.15c}$$

Therefore, any symmetric matrix A can be factored into the product of the diagonal matrix, Λ, containing the eigenvalues of the matrix which is pre- and post-multiplied by the orthogonal matrix, E, containing the eigenvectors (EOFs) of the matrix (15.15a). *Or, any symmetric matrix A can be reduced to a diagonal matrix, whose elements consist of its eigenvalues, by pre- and post-multiplying the A matrix with the orthogonal matrix containing its eigenvectors (this is called* **diagonalization***).*

Since the eigenvectors of any symmetrical matrix are orthogonal to one another, they can be combined with appropriate coefficients to produce any nonzero vector, **z** (where the order of the symmetric matrix is the same as that of the vector). That is, they can be used like any other orthogonal basis for interpolating, smoothing or any other purpose for which we have employed such functions previously. Specifically,

$$\begin{aligned} \mathbf{z} &= c_1\mathbf{e}_1 + c_2\mathbf{e}_2 + \ldots + c_n\mathbf{e}_n \\ &= \mathbf{EC}, \end{aligned} \tag{15.16a}$$

where **C** is the matrix containing the c_i constants.

Pre-multiplying (15.16a) through by $\mathbf{E}^{\mathrm{T}}$ gives

$$\begin{aligned} \mathbf{E}^{\mathrm{T}}\mathbf{z} &= \mathbf{E}^{\mathrm{T}}\mathbf{EC} \\ &= \mathbf{C}. \end{aligned} \tag{15.16b}$$

So, the coefficients in vector **C** are determined from $\mathbf{E}^{\mathrm{T}}\mathbf{z}$, from which **z** can be decomposed as **EC**. Thus the eigenvectors form an orthogonal basis for **z**, and *the details (i.e., structure or shape) of the basis changes with the composition of the matrix **A** from which it was derived.*

Alternatively, consider (15.15) for a zero-mean stationary/ergodic multidimensional data set,[12] containing a variable sampled at *n* locations and *m* times (containing an *m*-dimensional time series for each row), and stored in **data matrix X** whose sample covariance matrix $\hat{\Sigma}_{\mathbf{X}}$ is proportional to $\mathbf{XX}^{\mathbf{T}}$.

[12] The stationarity/ergodic restriction is imposed so that we can estimate the covariance matrix directly from the major product moment XXT. Otherwise, the covariance matrix is given as the expected value of the major product moment, as explained in Chapter 7. The standardization avoids the need for additional scaling of the major product moment to give variances and covariances.

$$\mathbf{X} = \overbrace{\begin{matrix} j=1 & j=2 & & j=m \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & & x_{2m} \\ & \vdots & & \\ x_{n1} & x_{n2} & & x_{nm} \end{bmatrix} & & & \end{matrix}}^{\text{times}} \begin{matrix} i=1 \\ i=2 \\ \\ i=n \end{matrix} \text{ locations} \tag{15.17a}$$

$$\frac{\mathbf{X}\mathbf{X}^{\mathrm{T}}}{n-1} = \hat{\Sigma}_X = \begin{bmatrix} s_1^2 & s_{12} & \cdots & s_{1n} \\ s_{21} & s_2^2 & & s_{2n} \\ & & \vdots & \\ s_{n1} & s_{n2} & \cdots & s_n^2 \end{bmatrix} \tag{15.17b}$$

Where s_i^2 represents the sample temporal variance of the $\mathbf{X}$ series over all sampled times at location 1, and s_{21} represents the sample covariance between the time series obtained at locations 2 and 1. As constructed here, each *column* of the data matrix $\mathbf{X}$ contains the values of $\mathbf{X}$ at a time t = 1,2, . . ., or *m* across all *n* spatial locations, the second column contains all observations at time 2, etc.[13] In other words, the time series occupy the rows (*i*th row contains the time series for the *i*th location), and each column contains a *map* of values for time j (a "map" because it shows the values of the variable x at multiple locations all at the same time).

By placing time series in rows in the data matrix X (each row a different location) as many of us prefer, the operational order of the sample covariance matrix is the same for a single time series vector, x, as for a data matrix: $E[\mathbf{x}\mathbf{x}^T]$ and $\mathbf{X}\mathbf{X}^T$ (see "Variance Covariance Matrix" in Appendix 1, §A.73).[14]

Rewriting (15.15b) using the sample covariance matrix of $\mathbf{X}$ for matrix $\mathbf{A}(\hat{\Sigma}_{\mathbf{X}})$ and substituting $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ for $\hat{\Sigma}_{\mathbf{X}}$ gives ($\mathbf{E}$, $\hat{\Sigma}_{\mathbf{X}}$ and $\mathbf{\Lambda}$ are all $n \times n$ matrices),

$$\mathbf{E}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{E} = \Lambda, \tag{15.18a}$$

and, defining

$$\mathbf{C}_{nm} = \mathbf{E}_{nn}^{\mathrm{T}}\mathbf{X}_{nm} \tag{15.18b}$$

$$\mathbf{X}_{nm} = \mathbf{E}_{nn}\mathbf{C}_{nm}, \tag{15.18c}$$

where $\mathbf{C}$ is now a fully populated ($n \times m$) matrix of coefficients. Equation (15.18c) states that the spatial series in $\mathbf{X}$ can be re-expressed as a linear sum of (spatial) eigenvectors as one map, j, per time: i = 1, *m*. Alternatively, in terms of the eigenvector maps and time-series coefficients (one complete time series for each map),

[13] Some authors define a data matrix so that the time variation is given as column vectors (versus their presentation as row vectors, here). This is fine if you prefer to take this approach, but in doing so, it is important to be clear which way the data matrix is being set up, so that all of the ensuing manipulations are done in the proper arrangement.

[14] Single vectors are column vectors, unless otherwise stated.

$$\mathbf{X} = \begin{bmatrix} & & & \\ \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \\ & & & \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1m} \\ c_{21} & c_{22} & c_{23} & & c_{2m} \\ \vdots & & & & \\ c_{n1} & c_{n2} & c_{n3} & \dots & c_{nm} \end{bmatrix}. \tag{15.18d}$$

Thus, the data are represented as a linear sum of the EOF basis, as described in (15.16a), and consistent with any orthogonal interpolation problem. In (15.16), it is clear that any vector can be described as a linear combination of *any* n-orthogonal eigenvectors; here, in (15.18), it is suggested that the eigenvectors arising from the sample covariance matrix of the data matrix may represent a natural basis for the data in question.

The coefficients, on the other hand, in (15.18b) look like

$$\mathbf{C} = \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \vdots \\ \mathbf{e}_n^T \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1m} \\ x_{21} & x_{22} & x_{23} & & x_{2m} \\ \vdots & & & & \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nm} \end{bmatrix}. \tag{15.18e}$$

So, each coefficient is simply the sum of the eigenvector elements with the corresponding element in $\mathbf{x}_j$ (i.e., the *projection* of the data on the eigenvectors). That is, it is the correlation between the eigenvector and data. That is exactly how the Fourier transform works. Consider the discrete Fourier transform, which gives the coefficient $\mathrm{J}_j = n^{-1}\sum_{p=1}^{n} x_p \mathrm{e}^{-i\omega_j t_p}$, where the eigenvector is the $\mathrm{e} - i\omega\mathrm{t}$ and each element of the data vector, $\mathbf{x}_j$, coincides with each time in x_t. Likewise, (15.18c) is of the same form as the inverse Fourier transform, converting the coefficients of the frequency domain (the orthogonal function domain) back to the original domain. Thus, the Fourier transform is just a specific case of the more general form that applies for any orthogonal basis, though as shown in (D15.1), the Fourier kernel ($\mathrm{e} - i\omega\mathrm{t}$) is the natural eigenfunction of linear systems.

Note two important properties (though there are many more) of eigenvalues: (1) the trace of matrix $\hat{\Sigma}_{\mathbf{X}} = \mathrm{tr}(\hat{\Sigma}_{\mathbf{X}})$ = the sum of the λ_i = sum of the total variance in the data matrix over all time and space locations and (2) the determinant of matrix $\hat{\Sigma}_{\mathbf{X}} = |\hat{\Sigma}_{\mathbf{X}}| = \prod \lambda_i$. Also, a singular matrix must have at least one nonzero eigenvalue, and in fact the number of nonzero eigenvalues is equal to the rank of the matrix.

Singular Value Decomposition (SVD) Approach

Consider the relationship shown in (15.18b,c). Actually solving for eigenvalues typically requires a matrix method such as singular value decomposition (**SVD**). Recall from (A.51), the relationship defining SVD, that any $n \times m$ matrix $\mathbf{A}$ ($n \geq m$) can be decomposed as

$$\mathbf{A}_{nm} = \mathbf{U}_{nn}\mathbf{S}_{nm}\mathbf{V}^{\mathrm{T}}_{mm}, \tag{15.19; A.51}$$

where $\mathbf{U}$, $\mathbf{V}^{\mathrm{T}}$ are orthogonal and $\mathbf{S}$ is a diagonal matrix. Therefore, if you perform SVD on your data matrix, $\mathbf{X}$, you obtain $\mathbf{X} = \mathbf{USV^T}$. From this you can form the sample covariance matrix, $\hat{\Sigma}_{\mathbf{X}}$, in the usual manner ($\mathbf{XX^T}$), allowing

$$\mathbf{XX}^{\mathrm{T}} = \mathbf{USV}^{\mathrm{T}}\mathbf{VS}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}} = \mathbf{USS}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}. \tag{15.20}$$

Calling $\mathbf{SS}^{\mathrm{T}}$, $\mathbf{\Lambda}$ (i.e., *the singular values in S are the square roots of the eigenvalues in* Λ), gives

$$\mathbf{XX}^{\mathrm{T}} = \mathbf{U}\Lambda\mathbf{U}^{\mathrm{T}}. \tag{15.21}$$

This is of the same form as (15.15a), $\mathbf{A} = \mathbf{E}\Lambda\mathbf{E}^{\mathrm{T}}$. It is a unique composition, thus: $\mathbf{SS}^{\mathrm{T}} = \mathbf{\Lambda}$ represents the eigenvalues of $\mathbf{XX}^{\mathrm{T}}$, and $\mathbf{U}$ (containing the **left singular vectors**) represent the eigenvectors (the $\mathbf{E}$ in (15.15a)) of $\mathbf{XX}^{\mathrm{T}}$, our sample covariance matrix.

Likewise, if we form our covariance matrix between space series (instead of between time series), then, $\hat{\Sigma}_{\mathbf{X}} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$, and a repeat of the above procedure shows that

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{VS}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{USV}^{\mathrm{T}} = \mathbf{VS}^{\mathrm{T}}\mathbf{SV}^{\mathrm{T}}. \tag{15.22}$$

So in this case, $\mathbf{V}$ (the **right singular vectors**) contain the eigenvectors of the covariance matrix formed as $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, and $\mathbf{S}^{\mathrm{T}}\mathbf{S}$ contain the eigenvalues.

Thus, we can apply SVD analysis directly to the original (non-square) data matrix, X, and obtain the eigenvectors and eigenvalues without ever having to actually construct the sample covariance matrix:

$$\mathbf{X}_{nm} = \mathbf{U}_{nn}\mathbf{S}_{nm}\mathbf{V}^{\mathrm{T}}_{mm}. \tag{15.23}$$

Furthermore, the SVD is a very robust decomposition method that typically provides a solution even if the covariance matrix is only marginally stable.

EOF Orthogonal Fitting of a Straight Line

In Chapter 5 (Smoothed Curve Fitting), the simple solution of the orthogonal fitting a straight line in a regression via EOFs was deferred to this chapter after explanation of the fundamentals of the EOF methodology. In §5.6 a more complicated approach was given where you rotate the axes to explain the most variance. In this section, the angle θ ($\hat{\theta}$ the amount remaining in §5.6) is found via EOF analysis. Here, I provide the solution as shown by Jackson (1991).

First, place your n (x,y) data pairs in a $n \times 2$ matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \\ x_n & y_n \end{bmatrix}. \tag{15.24}$$

Now perform a singular-value decomposition on the X matrix, giving

$$\mathbf{X}_{nm} = \mathbf{U}_{nn}\mathbf{S}_{nm}\mathbf{V}^{\mathrm{T}}_{mm}. \tag{15.25}$$

The columns of $\mathbf{V}$ contain the eigenvectors ($\mathbf{e_1}$, $\mathbf{e_2}$) of the system, in this case, m = 2.

$$\mathbf{V} = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} \tag{15.26}$$

Now compute the direction cosines associated with each element of $\mathbf{e_1}$. So,

$$cos\, \mathrm{e}_{11} = \theta_{11}. \tag{15.27}$$

θ_{11} gives the angle of rotation from the abscissa, and θ_{21} the angle of rotation from the ordinate ($\theta_{11} + \theta_{21} = 90°$). The orthogonal best-fit line is the line with the θ_{11} rotation.

15.4 Principal Components (PC)

Principal Components Analysis (**PCA**) is a specific case of EOF analysis and is one of the classic methods of multivariate statistical analysis. There is considerable material written about it in the multivariate statistics texts (e.g., Cooley and Lohnes, 1971; Anderson, 1984). It involves particular linear combinations of the original random variables such that they are orthogonal to one another and represent the greatest amount of variance (of the original set) in the fewest number of terms.[15] Thus, if we have p variables, the principal components are p new variables that sum together to interpolate the original data set, with the special characteristics that the new p variables are not correlated to one another, the first variable describes the most variance of the original dataset, the second variable describes the next-most variance (i.e., the most variance of the remaining undescribed variance), etc. Recall that in spectral analysis, the power of each fitted cosine was proportional to the amount of variance in the original time series that it described. The same is true here (as will be shown), only in this case the new variables are ordered so that each one describes less variance than the preceding ones. In this manner, it is hoped that we might find some small subset of new variables that describe most of the original variance in the data set, and that we only need to retain this minimal number of new variables (principal components) to retain the series signal.

For example, if we have measured, over some region of the ocean, the surface temperature, salinity, CO_2, freons, He^3, and eight other tracers, we might expect that the distribution of several of these are related to one another (i.e., are distributed via the same physical processes and have similar source/sink functions). Consequently, we might expect that some linear combination of the related variables will describe a significant amount of the variance displayed by the entire dataset, say 50 percent of the observed variance. Another combination may describe another large fraction of the variance, say 40 percent, while the remaining 11 combinations each describe only some

[15] These represent the different variables that are related through a joint probability distribution, or in terms of the physical problem of interest, related through the phenomenon being studied.

small fraction (that we hope to be noise). Therefore, by retaining just the first two principal components, we are describing 90 percent of the total variance through only two orthogonal composite variables, instead of the original bulky 13 variables. Those variables were not independent, and thus were dragging along a lot of redundant information. Furthermore, since these two principal components are orthogonal, they may each represent different aspects of the local physics in the area, and by working strictly with them we might isolate the two physical processes that seem to dominate the distribution of all of these parameters in the region.[16]

Finally, we need not make any assumptions about the nature of the multivariate distribution between the various variables we will be examining. However, if the variables do obey a multivariate normal distribution, then they can be completely described by their means and variance-covariance structure (i.e., via a covariance matrix). Since the method of principal components makes use of this covariance matrix, for multivariate normal data, given zero mean (or normalizing as such), we are completely explaining the joint distribution of the variables through this analysis.

15.4.1 Definitions

Consider defining n new random variables, P_l, as the linear combination, or composite, of n original random variables (or locations), each variable contained within a vector $\mathbf{X}_i$.[17] Written in terms of random variables, P_l and X_i (i.e., not in terms of vectors, though each random variable can also be thought of as a vector),

$$\begin{aligned} P_1 &= a_{11}X_1 + a_{12}X_2 + \ldots + a_{1n}X_m \\ P_2 &= a_{21}X_1 + a_{22}X_2 + \ldots + a_{2n}X_m \\ &\vdots \\ P_n &= a_{n1}X_1 + a_{n2}X_2 + \ldots + a_{nn}X_m. \end{aligned} \tag{15.28}$$

Each of these P_i is called a **principal component**. If the data represent a space-time field so that at different locations we have measurements at some specific times, then it is easy to see that P_1 is the x value estimated to exist at one of the locations, given its multivariate correlation to the values taken at the same time at all of the other n space locations. In other words, we are saying that the value at any one location is correlated to the values observed at the other locations (the X_i), and we might use that covarying information to predict a single composite time series containing all of the covarying information. That is, *a principal component is a composite variable formed by simple multivariate linear regression that contains in this one composite time series all of the shared information, eliminating the need for all of the covarying time series carrying*

[16] Often, factor analysis, another variant of EOF analyses, is more appropriate for this particular purpose.

[17] This is essentially the reverse operation of what we did previously with the eigenvectors, in which case we combined eigenvectors to reproduce the data (15.18c). Here, we wish to construct p new variables as linear combinations of the original data. However, it is easily seen that the principal components defined here are comparable to the coefficients arising from the multiplication of the data matrix with the eigenvector matrix, shown in (15.18b).

replicate information. The various time elements in each principal component are given as

$$\begin{aligned} p_{\ell 1} &= a_{\ell 1}x_{11} + a_{\ell 2}x_{21} + \ldots + a_{\ell n}x_{n1} \\ p_{\ell 2} &= a_{\ell 1}x_{12} + a_{\ell 2}x_{22} + \ldots + a_{\ell n}x_{n2} \\ &\vdots \\ p_{\ell n} &= a_{\ell 1}x_{1m} + a_{\ell 2}x_{2m} + \ldots + a_{\ell n}x_{nm} \end{aligned}$$

$$p_{\ell j} = \sum_{k=1}^{n} a_{\ell k}x_{kj}. \tag{15.29a}$$

In other words, the weights used to linearly combine each random variable change only with the random variable, not with the value of the random variable.

In matrix notation,

$$\begin{aligned} \mathbf{p}_1^T &= \mathbf{a}_1^T\mathbf{X} \\ \mathbf{p}_2^T &= \mathbf{a}_2^T\mathbf{X} \\ &\vdots \\ \mathbf{p}_n^T &= \mathbf{a}_n^T\mathbf{X}, \end{aligned} \tag{15.29b}$$

or

$$\mathbf{P} = \mathbf{A}\mathbf{X}, \tag{15.29c}$$

where $\mathbf{p}_i$, $\mathbf{a}_i$ and $\mathbf{x}_i$ are column vectors, with $\mathbf{p_i^T} = [p_{i1}\ p_{i2} \ldots p_{in}]$, $\mathbf{a_i^T} = [a_{i1}\ a_{i2} \ldots a_{in}]$ and $\mathbf{x_i^T} = [x_{i1}\ x_{i2} \ldots x_{in}]$, **P, A** and **X** are matrices with each row, i, containing the $\mathbf{p}_i^T$, $\mathbf{a}_i^T$ and $\mathbf{x}_i^T$.

Therefore, if the *n* data variables are u_i, $v_i \ldots$, z_i, then in matrix form,

$$\mathbf{X} = \begin{bmatrix} u_1 & u_2 & & u_m \\ v_1 & v_2 & \ldots & v_m \\ \vdots & & & \\ z_1 & z_2 & & z_m \end{bmatrix},$$

where each row represents a different variable and each column represents the time at which that variable was measured. Alternatively, if the data consist of only one variable, say temperature as a function of spatial location, then we might represent the data as before (*m* times and *n* spatial locations),

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & & x_{nm} \end{bmatrix},$$

where each column is again time, but this time the rows represent different spatial locations. In either case, the data matrix can be represented via time-synchronous column vectors (each *element* in a vector $\mathbf{f}_j$ representing a different spatial location (= map) or a different variable, but always measured at the same times (j = 1 − *m*), in this example where time is the independent variable of interest), as

$$\mathbf{X} = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_m \end{bmatrix}. \tag{15.29d}$$

For any of the representations of the data matrix, (15.29) can be written in terms of the principal components, i.e., the $\mathbf{P}_\ell$ vectors, as[18]

$$\mathbf{P}_\ell = a_{\ell 1}\mathbf{X_1} + a_{\ell 2}\mathbf{X_2} + \ldots + a_{\ell n}\mathbf{X}_n = \mathbf{X}^{\mathrm{T}}\mathbf{a}_\ell. \tag{15.30a}$$

Or,

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_n^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} \begin{bmatrix} u_1 & u_2 & & u_m \\ v_1 & v_2 & \dots & v_m \\ \vdots & & & \\ z_1 & z_2 & & z_m \end{bmatrix} \quad n \text{ variables over } m \text{ times}$$

$$= \begin{bmatrix} a_{11} & a_{12} & & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & & x_{nm} \end{bmatrix} \quad \begin{array}{l} 1 \text{ variable measured at} \\ n \text{ locations over } m\text{times} \end{array}$$

$$= \begin{bmatrix} a_{11} & a_{12} & & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_m \end{bmatrix} \quad \begin{array}{l} n - \text{length vectors} \\ \text{measured over } m \text{ times} \end{array}$$

which is seen by examining the elements of the ℓ th PC (each element representing a different time, j),

$$\mathbf{p}_\ell = \begin{bmatrix} p_{\ell 1} \\ p_{\ell 2} \\ p_{\ell 3} \\ \vdots \\ p_{\ell m} \end{bmatrix} = a_{\ell 1} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ \vdots \\ x_{1m} \end{bmatrix} + a_{\ell 2} \begin{bmatrix} x_{21} \\ x_{22} \\ x_{23} \\ \vdots \\ x_{2m} \end{bmatrix} + \ldots + a_{\ell n} \begin{bmatrix} x_{n1} \\ x_{n2} \\ x_{n3} \\ \vdots \\ x_{nm} \end{bmatrix}, \tag{15.30b}$$

[18] This is a direct application of the reversal rule of transposed products to (15.29b). That is, if $p^{\mathrm{T}} = a^{\mathrm{T}}x$, then the transpose of this is $p = (a^{\mathrm{T}}x)^{\mathrm{T}} = x^{\mathrm{T}}a$.

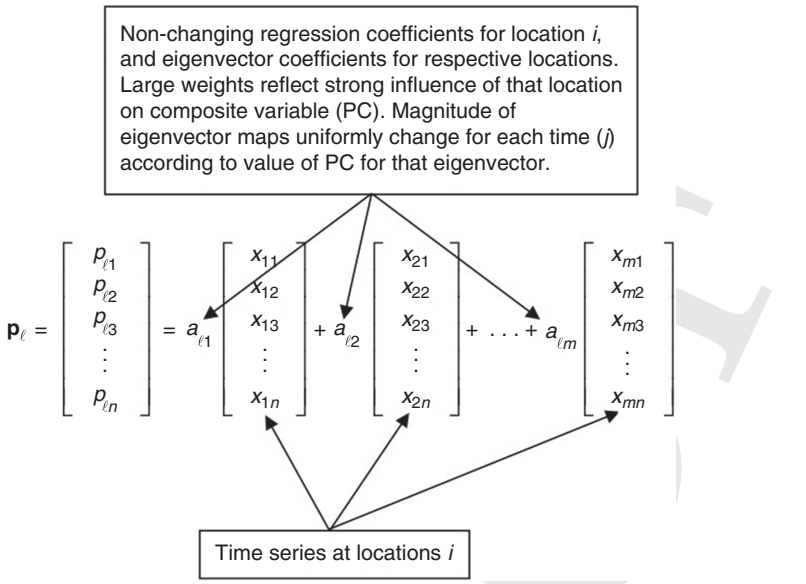


Figure 15.2 Description of each part of Principal Component, l. From this, it is clear that the PC has the same units as the x variable.

directly giving the form of (15.30a) and (15.29a). In terms of multivariate series, u, v, . . ., z, (15.30b) is

$$\mathbf{p}_i = \begin{bmatrix} p_{i1} \\ p_{i2} \\ p_{i3} \\ \vdots \\ p_{im} \end{bmatrix} = a_{i1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_m \end{bmatrix} + a_{i2} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_m \end{bmatrix} + \ldots + a_{in} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_m \end{bmatrix}. \tag{15.30c}$$

So, each principal component is an m-length vector containing the weighted linear sum of the n variables in the data set at each of m times. The weighting applied to each variable to form the new composite variable, the principal component, doesn't change with time as shown in Figure 15.2 (i.e., the variables are added as a weighted sum to form the principal component at time t = j = 1, using the same weights used to form the principal component at time t = j = m), so the weights are static. *This is a multivariate regression problem: the p_i are predicted as linearly related to the various x_i values. However, the principal components change in time, since the x_i variables themselves change with each time increment, even though the weights do not. The regression reduces the interdependencies between the variables to new variables that contain the interdependencies as the weighted sum in a single new variable.*

15.4.2 Solving for the PC Coefficients

Constrained Regression

We wish to determine those coefficient values in matrix **A** of (15.29c) such that the principal components in **P** sequentially describe the maximum amount of variance in **X** or its residual (i.e., that variance not described by the previous PCs). That is, the new variables, the PCs, are to be constructed so that they are uncorrelated to one another; the first one describes a maximum amount of variance in the original dataset; the second PC describes the most amount of variance in the data that were not described by the first PC (i.e., the residual variance). Each PC describes the maximum amount of the remaining variance until eventually the *n*th PC describes the final residual so that the *n* PCs together describe all of the variance in **X**. This solution is accomplished via constrained regression using the method of Lagrange multipliers, as presented in Chapter 6.

First, consider the variance of the principal components, P_i (which measures *variance in time, for the data matrix as defined here*). This is done directly from the formula developed earlier for the variance of a general linear function, where if

$$P_i = y(X_i) = \sum_{i=1}^{m} a_i X_i \tag{15.31a}$$

$$\mathbf{p}_i = \mathbf{X}^T \mathbf{a}_i, \tag{15.31b}$$

then, according to (2.53a),

$$\mathrm{Var}[y(X)] = \sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j \mathrm{Cov}[X_i X_j], \tag{15.32a}$$

or in matrix form,

$$= \mathbf{a}^T \boldsymbol{\Sigma} \mathbf{a}, \tag{15.32b}$$

where $\boldsymbol{\Sigma}$ is the (true) covariance matrix of X.[19]

Thus, for the *i*th PC,

$$\mathrm{Var}[P_i] = \mathbf{a}_i^T \boldsymbol{\Sigma} \mathbf{a}_i. \tag{15.33a}$$

Similarly, the covariance between the different P_i is given as

$$\mathrm{Cov}[P_i P_j] = \mathbf{a}_i^T \boldsymbol{\Sigma}_{ij} \mathbf{a}_j. \tag{15.33b}$$

For principal components analysis, we wish to determine the a_{ij} coefficients in **A** such that $\mathrm{Cov}[P_i P_j] = 0$ when $i \neq j$ (i.e., making the principal components uncorrelated), and such that the variance described by P_1 is greater than that described by P_2, or any of the other PCs. The *i* th principal component describes the *i* th-largest variance, so $\mathrm{Var}[P_1] >$

[19] This will be estimated by the sample covariance matrix, $\hat{\Sigma}_{\mathbf{x}}$.

$\mathrm{Var}[P_2] > \mathrm{Var}[P_3]$, etc. We wish to maximize $\mathrm{Var}[P_1]$, or $\mathbf{a}_1^{\mathrm{T}} \sum \mathbf{a}_1 = 1$. Also, in order to preserve the total variance of the original system, we must normalize the coefficients such that their sum of squares (i.e., their scaling of the variance) is $\mathbf{a}_1^{\mathrm{T}}\mathbf{a} = 1$.

First PC

We can determine the coefficients of the first principal component, $a_{1\ j}$, via the same techniques we have previously used. Specifically, we wish to find those coefficients that maximize the variance, $\mathbf{a}_1^{\mathrm{T}} \sum \mathbf{a}_1$, subject to the constraint that $\mathbf{a}_1^{\mathrm{T}}\mathbf{a} - 1 = 0$.[20] This is done using the method of Lagrange multipliers, where the Lagrangian equation (to be maximized with the constraint) is given as[21]

$$\begin{aligned} \mathrm{L}(\mathbf{a}_1) &= \mathrm{f}(\mathbf{a}_1) - \lambda\theta(\mathbf{a}_1) \\ &= \mathbf{a}_1^{\mathrm{T}}\Sigma\mathbf{a}_1 - \lambda(\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 - 1) \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n} a_{1i}a_{1j}\sigma_{ij} - \lambda\left(\sum_{j=1}^{n} a_{1j}^2 - 1\right). \end{aligned} \tag{15.34}$$

At the maximum, the derivative of the Lagrangian with respect to the unknown coefficient is zero,

$$\frac{\partial L}{\partial a_1} = \frac{\partial}{\partial a_{1j}}\left(\sum_{j=1}^{n}\sum_{i=1}^{n} a_{1i}a_{1j}\sigma_{ij}\right) - \lambda\frac{\partial}{\partial a_{1j}}\left(\sum_{j=1}^{n} a_{1j}^2 - 1\right) = 0, \tag{15.35a}$$

which, recalling that $\partial(\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x})/\partial\mathbf{x} = 2\mathbf{A}\mathbf{x}$, reduces to

$$2\Sigma\mathbf{a}_1 - 2\boldsymbol{\lambda}\mathbf{a}_1 = 0 \tag{15.35b}$$

or

$$(\Sigma - \boldsymbol{\lambda}I)\mathbf{a}_1 = 0. \tag{15.35c}$$

What Great Luck!![22] *This potential mess just reduced to the standard form of the eigenvalue problem given in* (15.1b)*, where the eigenvalues correspond to the covariance matrix, Σ.* That is, we can use this equation to solve for the Lagrange multiplier, or now clearly the eigenvalue λ, since the nontrivial solution only exists by making $(\boldsymbol{\Sigma} - \lambda\mathrm{I})$ singular.

The covariance matrix $\boldsymbol{\Sigma}$ must be estimated from the data via the sample covariance matrix, $\hat{\Sigma}_X = \mathbf{X}\mathbf{X}^{\mathrm{T}}$, and the eigenvalues determined from this, so the determinant is

$$|\hat{\Sigma}_x - \boldsymbol{\lambda}I|\mathbf{a}_1 = 0, \tag{15.36}$$

[20] This constraint, that the coefficients conserve variance, is simply that $\mathbf{a}_1^{\mathrm{T}}\mathbf{a} = 1$, but rewritten as required by the method of Lagrange multipliers: that is, $g(a) = 0$, or $\mathbf{a}_1^{\mathrm{T}}\mathbf{a} - 1 = 0$.

[21] In anticipation of results soon to follow, we define the Lagrange multiplier as $-\lambda$. Since this is an arbitrary constant to be solved for, this does nothing more than add later convenience.

[22] Or was it magic?

which is a polynomial in λ of order n, with n roots. Since $\hat{\Sigma}_{\mathbf{X}}$ is positive definite, these roots, the eigenvalues, are all real and positive, and we order them such that

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n \,. \tag{15.37}$$

We can now introduce the constraint equation to complete the constrained maximization problem by multiplying (15.35b) by $\mathbf{a}_1^{\mathrm{T}}$ to give

$$\mathbf{a}_1^{\mathrm{T}}\hat{\Sigma}_X\mathbf{a}_1 \; - \; \lambda\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 \; = \; 0. \tag{15.38a}$$

Since $\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 = 1$, introduction of this constraint gives

$$\mathbf{a}_1^{\mathrm{T}}\hat{\Sigma}_X\mathbf{a}_{11} \; = \; \lambda. \tag{15.38b}$$

As shown in (15.33a), $\mathrm{Var}[P_1] = \mathbf{a}_1^{\mathrm{T}}\Sigma\mathbf{a}_1$, and since (15.38b) shows that $\mathbf{a}_1^{\mathrm{T}}\hat{\Sigma}_X\mathbf{a}_{11} \; = \; \lambda$ then,

$$\mathrm{Var}_{sample}[\mathrm{P}_1] \; = \; \lambda. \tag{15.39}$$

The most variance is described by λ_1, since we ordered the eigenvalues in decreasing size (15.37). Therefore, the constraint effectively informs us to use the first eigenvalue in order to maximize the variance, though it explicitly forces us to work with normalized eigenvectors. So the first principal component, that which describes the maximum variance of linear combinations of the original variables, is given by

$$\mathbf{p}_1^{\mathrm{T}} \; = \; \mathbf{a}_1^{\mathrm{T}}\mathbf{X}, \tag{15.40}$$

where a_1 is that (normalized) eigenvector that satisfies $(\hat{\Sigma}_{\mathbf{X}} - \lambda_1\mathbf{I})\mathbf{a}_1 \; = \; 0$. That is, the *coefficients* required to make this new composite variable, or PC, is in fact given by the first *eigenvector* of $\hat{\Sigma}_{\mathbf{X}}$.

Second PC

For the second principal component, we now wish to find that coefficient vector (i.e., those values of $\mathbf{a}_2^{\mathrm{T}}$) that maximizes the amount of remaining variance undescribed by the first principal component. The original constraint that the weights conserve variance must now be subjected to an additional constraint, that this principal component is uncorrelated to the first one. So,

$$max\{\mathrm{Var}[\mathrm{P}_2](a_2)\} \; = \; max\{\mathbf{a}_2^{\mathrm{T}}\Sigma\mathbf{a}_2\}, \tag{15.41a}$$

subject to

$$\mathbf{a}_2^{\mathrm{T}}\mathbf{a}_2 \; = \; 1 \tag{15.41b}$$

$$\mathrm{Cov}[\mathrm{P}_1\mathrm{P}_2] \; = \; \mathbf{a}_2^{\mathrm{T}}\Sigma_{12}\mathbf{a}_1 \; = \; 0. \tag{15.41c}$$

Here, the matrix form of Cov[P_1P_2] in (15.41c) is found by expanding the sample covariance sum in terms of these two components and substituting in the vector expression from (15.29b):

$$\begin{aligned}\mathrm{Cov}[\mathbf{P}_1\mathbf{P}_2] = \hat{\Sigma}_{\mathbf{P}_{12}} &= \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{P}_{2i}\mathbf{P}_{1i} \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{a}_2^{\mathrm{T}}\mathbf{X}_i)(\mathbf{a}_1^{\mathrm{T}}\mathbf{X}_i)^{\mathrm{T}} \\ &= \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{a}_2^{\mathrm{T}}\mathbf{X}_i\mathbf{X}_i^{\mathrm{T}}\mathbf{a}_1 \\ &= \mathbf{a}_2^{\mathrm{T}}\hat{\Sigma}_{\mathbf{X}}\mathbf{a}_1.\end{aligned} \tag{15.42}$$

Further manipulation of this product shows that it is equivalent to $\mathbf{a}_2^{\mathrm{T}}\mathbf{a}_1\lambda_1$, and thus, since this is equal to zero, reconfirms that the eigenvectors must be uncorrelated to one another.[23]

Again we resort to constrained maximization using two Lagrange multipliers (λ and κ, the latter given as 2κ for convenience) to maximize the Lagrangian equation:

$$\begin{aligned}L(\mathbf{a}_2) &= f(\mathbf{a}_2) - \lambda\theta(\mathbf{a}_2) - 2\kappa\theta(\mathbf{a}_2, \mathbf{a}_1) \\ &= \mathbf{a}_2^{\mathrm{T}}\Sigma\mathbf{a}_2 - \lambda(\mathbf{a}_2^{\mathrm{T}}\mathbf{a}_2 - 1) - 2\kappa\mathbf{a}_2^{\mathrm{T}}\Sigma\mathbf{a}_1 \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n}a_{2i}a_{2j}\sigma_{ij} - \lambda\left(\sum_{j=1}^{n}a_{2j}^2 - 1\right) - 2\kappa\sum_{j=1}^{n}\sum_{i=1}^{n}a_{2i}a_{1j}\sigma_{ij}.\end{aligned} \tag{15.43}$$

At the maximum,

$$\frac{\partial L}{\partial \mathbf{a}} = \frac{\partial}{\partial a_{2j}}\left(\sum_{j=1}^{n}\sum_{i=1}^{n}a_{2i}a_{2j}\sigma_{ij}\right) - \lambda\frac{\partial}{\partial a_{2j}}\left(\sum_{j=1}^{n}a_{2j}^2 - 1\right) - 2\kappa\frac{\partial}{\partial a_{2j}}\left(\sum_{j=1}^{n}\sum_{i=1}^{n}a_{2i}a_{1j}\sigma_{ij}\right) = 0, \tag{15.44a}$$

which reduces to

$$2\Sigma\mathbf{a}_2 - 2\lambda\mathbf{a}_2 - 2\kappa\Sigma\mathbf{a}_1 = 0. \tag{15.44b}$$

Pre-multiplying through by $\mathbf{a}_1^{\mathrm{T}}$ gives

$$\mathbf{a}_1^{\mathrm{T}}\Sigma\mathbf{a}_2 - \lambda\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_2 - \kappa\mathbf{a}_1^{\mathrm{T}}\Sigma\mathbf{a}_1 = 0, \tag{15.44c}$$

allowing application of the second constraint equation, (15.41c) which indicates that the first two terms are zero (see (15.33) and the statement immediately below it). Also, we know already from (15.38b) that the third term in (15.44c), $\mathbf{a}_1^{\mathrm{T}}\hat{\Sigma}_{\mathbf{X}}\mathbf{a}_1 = \lambda_1$, is nonzero, so the only way (15.44c) can be satisfied, i.e., that

[23] This result comes almost directly from application of (15.15b), with the minor difference that here we need only examine one vector within the eigenvector matrix, E.

$$\kappa\lambda_1 = 0 \tag{15.44d}$$

is if $\kappa = 0$.

Finally, multiply (15.44b) by $\mathbf{a}_2^{\mathrm{T}}$, after setting $\kappa = 0$, to give

$$\mathbf{a}_2^{\mathrm{T}}\Sigma\mathbf{a}_2 - \lambda\mathbf{a}_2^{\mathrm{T}}\mathbf{a}_2 = 0. \tag{15.44e}$$

Application of the first constraint (second constraint has already been applied above), that $\mathbf{a}_2^{\mathrm{T}}\mathbf{a}_2 = 1$, yields

$$\mathbf{a}_2^{\mathrm{T}}\Sigma\mathbf{a}_2 = \lambda. \tag{15.44f}$$

As before, the variance of P_2 is equal to the above product, and thus λ_2 is that eigenvalue (the variance of P_2) that describes the maximum amount of as-yet undescribed variance. Therefore, as with the first principal component, the second one is simply determined in an analogous manner, where

$$\mathbf{p}_2^{\mathrm{T}} = \mathbf{a}_2^{\mathrm{T}}\mathbf{X}, \tag{15.45}$$

with $\mathbf{a}_2^{\mathrm{T}}$ being that normalized eigenvector that coincides with the λ_2 eigenvalue.

General Solution

This procedure is continued for each of the principal components, yielding each as simply products of the eigenvectors of the sample covariance matrix constructed from the original data matrix. The complete set of n principal components is represented as

$$\mathbf{P} = \mathbf{E}^{\mathrm{T}}\mathbf{X} \tag{15.46a}$$

or

$$\mathbf{p}_\ell^{\mathrm{T}} = \sum_{j=1}^{m} \mathbf{f}_j\mathbf{e}_\ell^{\mathrm{T}}, \tag{15.46b}$$

where the eigenvectors in $\mathbf{E}$ are derived from the sample covariance matrix, $\mathbf{X}\mathbf{X}^{\mathrm{T}}/(n-1)$. So,

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_n^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \vdots & & & \\ e_{n1} & e_{n2} & & e_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & & x_{nm} \end{bmatrix}, \tag{15.46c}$$

where the rows of the coefficient matrix (**e**, or **a** in (15.36)), are the eigenvectors, $\mathbf{e}_l$, derived from the covariance matrix,

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_n^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1^{\mathrm{T}} \\ \mathbf{e}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{e}_n^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & & x_{nm} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1^{\mathrm{T}} \\ \mathbf{e}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{e}_n^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_m \end{bmatrix}. \tag{15.46d}$$

Here (15.46d) *is identical to* (15.18b)*, but now we see that the coefficient vectors in C are the principal components, P.*

15.4.3 Interpretation and Use

Since *the variance of each principal component is equal to its corresponding eigenvalue, the total sample variance of the original data set*, s_x^2*, is equal to the sum of the eigenvalues*, and thus equal to the total variance of the principal components, i.e., $s_X^2 = \mathrm{tr}(\hat{\Sigma}_X) = \mathrm{tr}(\Lambda)$. The relative fraction of variance described by each principal component is equal to $\mathrm{Var}[P_i]/\mathrm{tr}(\Lambda)$, or λ_i/s_X^2.

The principal components are nothing more than a new system of independent (uncorrelated) vectors that describe the complete dataset in the EOF domain, but in such a manner that most of the variance of the data is now contained in the first few principal components. Such a system is demonstrated in Figure 15.3. The similarity between the functions in (15.18d) and (15.46b), or their matrix equivalents in (15.18c) and (15.46a), are analogous to Fourier transform pairs. That is, (15.18c) acts like an inverse Fourier transform in which the original series can be recovered, given knowledge of the coefficients of the orthogonal basis function (i.e., this is the interpolation of the original data set, given the coefficients to the orthogonal interpolant). Equation (15.46b) serves the role of a Fourier transform, which converts the data from the standard time (or other, e.g., space) domain to the new, EOF domain. In a Fourier transform, the new domain was the frequency domain, given the EOFs that are harmonic sines and cosines of frequency f_i.

In this case, the analysis found the shape based on the data (i.e., didn't assume cycles) that shared a common time history. The shape sharing this common time history is the eigenvector (the EOF) and how its magnitude varies in time is given by the PC (multiply the eigenvector for any particular time by the value of the PC at that time, giving a map at that time, scaled to capture the shape at that time).

Note that the principal components, i.e., the vectors in **P**, are orthogonal to each other, hence from (15.18a), $\mathbf{P}\mathbf{P}^{\mathrm{T}} = \Lambda$, so they have completely eliminated any correlation between transformed variables. That is, both the eigenvectors and their time-varying coefficients, the PCs, are *orthogonal* sets.

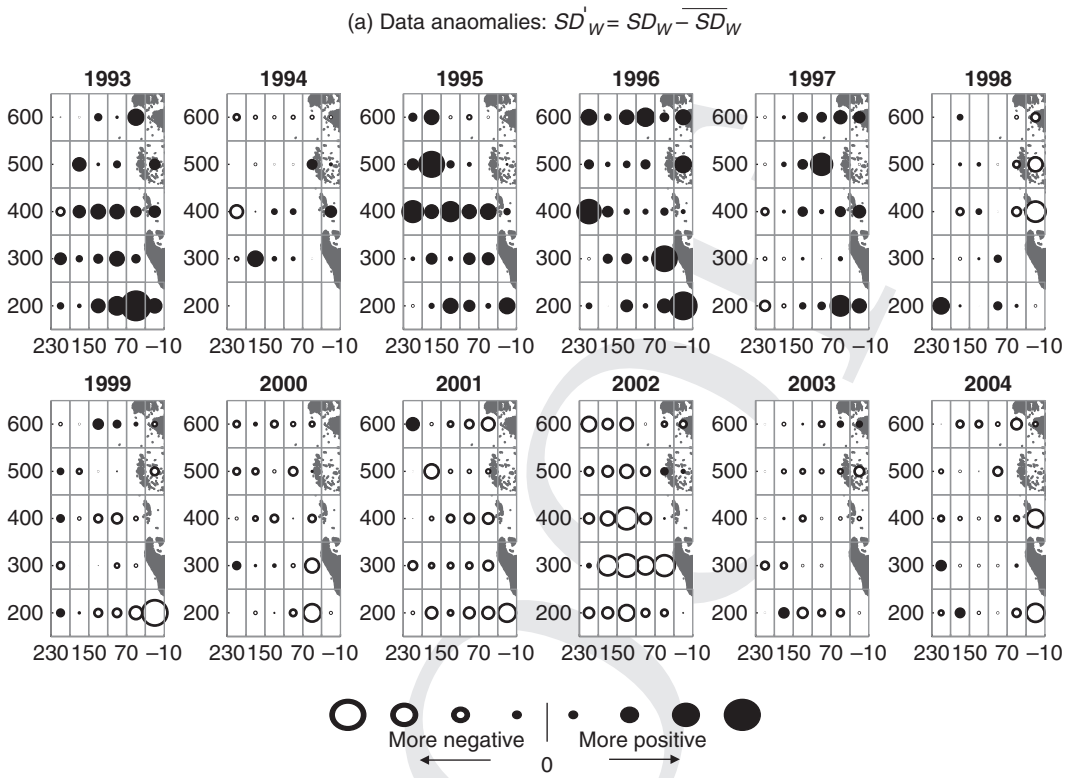


Figure 15.3 For an example of Figure 15.1, **(A)** anomaly maps for salt deficit, SD'_W ; **(B)** here are the first four modes of the EOF analysis. The first mode describes ~51 percent of the total temporal variance (explaining why changes in PC1 were apparent from visual inspection of the anomaly). Second mode still captures a fair amount of variance, but third and fourth modes less so – in fact, according to North's rule of thumb (discussed below), these two modes are too similar to separate and should be combined into a single mode.

Variance Decomposition

One can now consider the principal components in terms of their description of the variance in the original data, stored in **X**. That is, from (15.15a) and (15.18a) we see that the variance of the data, as described in the covariance matrix, **Σ**, or the sample covariance matrix, $\hat{\Sigma}_{\mathbf{X}}$, can be completely described by the eigenvectors and eigenvalue matrix as

$$\mathbf{X}\mathbf{X}^{\mathrm{T}} = \hat{\Sigma}_{\mathbf{X}} = \mathbf{S} = \mathbf{E}\Lambda\mathbf{E}^{\mathrm{T}}, \tag{15.47a}$$

or, in summation form,

$$\begin{aligned} &= \sum_{i=1}^{n} \lambda_i \mathbf{e}_i \mathbf{e}_i^{\mathrm{T}} \\ &= \lambda_1 \mathrm{e}_1 \mathrm{e}_1^{\mathrm{T}} + \lambda_2 \mathrm{e}_2 \mathrm{e}_2^{\mathrm{T}} + \ldots + \lambda_n \mathrm{e}_n \mathrm{e}_n^{\mathrm{T}}, \end{aligned}$$

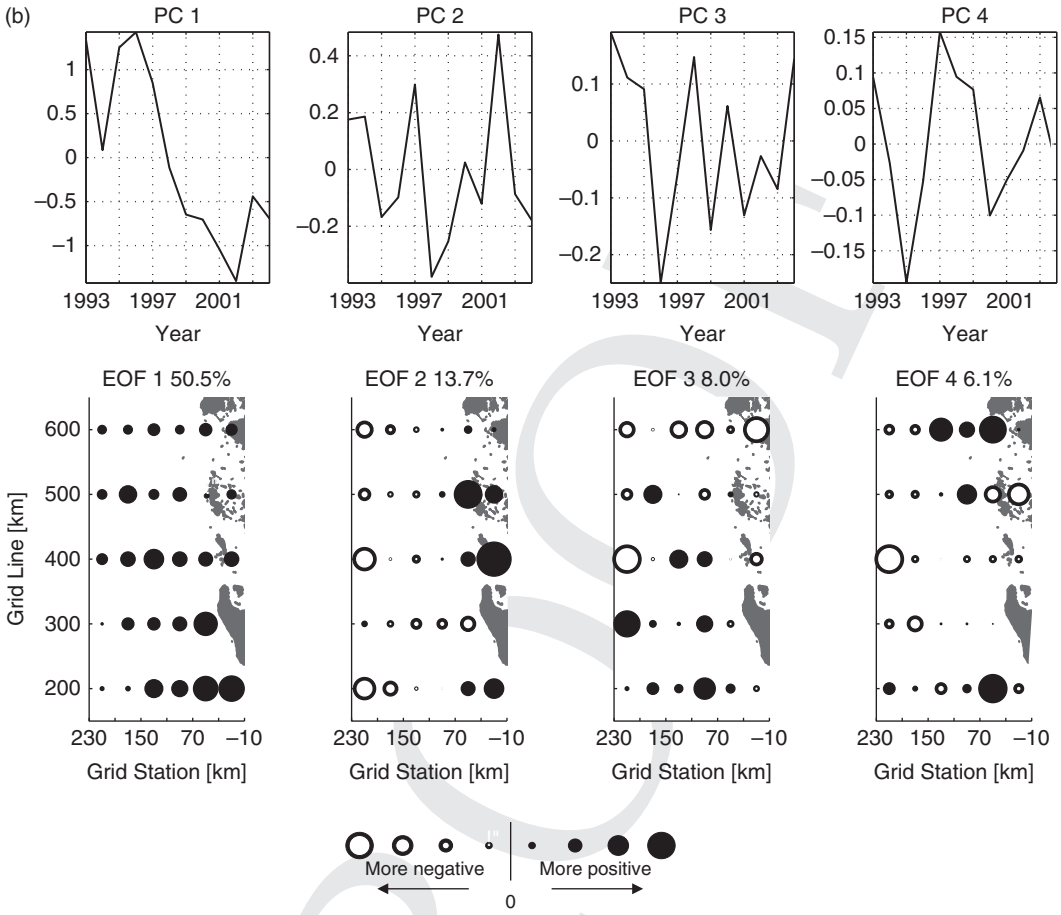


Figure 15.3 (cont.)

the sum of outer products, each giving a matrix allowing

$$= \mathbf{S}_1 + \mathbf{S}_2 + \ldots + \mathbf{S}_n \tag{15.47b}$$

$$\text{where, } \mathbf{S}_i = \lambda_i \mathbf{e}_i \mathbf{e}_i^{\mathrm{T}}.$$

Therefore, this same variance can be described (factored) as orthogonal, and thus additive, submatrices, $\mathbf{S}_i$, where each submatrix is the covariance structure of the particular eigenvector.

Equation (15.47) is known as a **spectral decomposition** of a matrix **A**.

If we determine the minimum number of components, p, that can be used to describe the maximum amount of variance in the dataset, then (15.47) can be written in terms of the sum of those components that contain most of the variance of the data, known as the **component theory, signal** or **information matrix**, $\hat{\mathbf{S}}$ and the sum of those remaining

$n - p$ components describe little variance, and thus represent the **residual matrix, error** or **noise matrix**, $\widetilde{\mathbf{S}}$. In this case, the total variance is now described as

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{\mathbf{x}} &= \hat{\mathbf{S}} + \widetilde{\mathbf{S}} \\ \mathbf{S}_i &= \lambda_i \mathbf{e}_i \mathbf{e}_i^{\mathrm{T}} \end{aligned} \tag{15.48}$$

So, we are stating that the EOF basis determined, given the principal component approach, can be reduced from n vectors to p vectors with minimal loss of signal. That is, we have found those eigenvectors that maximize the variance in the least number of basis vectors.

How many modes (EOFs plus their principal components) you keep can be estimated in a variety of ways. One of the most general rules of thumb is to only keep those that describe more than $1/n$ of the total variance. That is, if there are 100 variables, then if the variance was evenly distributed throughout all of them, each variable would explain 1 percent of the total variance. After transforming to principal components, we only wish to preserve those components that explain more than the average variable would. This rule is typically prone to err on the side of keeping too many coefficients.

Alternatively, you may make use of a **scree plot**, in which the fraction of variance explained by each component is plotted as a function of the component. This is exactly analogous to making a power spectrum plot in spectral analysis, only in this case one only retains those components that lie above some obvious background "flat" level (or "noise floor") that the higher-order terms display. This test works well in those situations where the transition from the information matrix to the noise matrix is rather abrupt.

Finally, you can apply a more rigorous test in the form of a sphericity test. In this case, after extracting each eigenvalue the test checks to see if the remaining, unexplained covariance matrix is significantly different from singular. That is, is the determinant indistinguishable from zero? (Recall that the determinant is equal to the sum of the remaining eigenvalues.)

You can perform a similar type of test initially, before you do the principal components analysis, to determine whether the variables are not already orthogonal to one another. In this case, the test is to see if the original covariance matrix is significantly different from the identity matrix. If not, then the variables are already as independent as they can get, and additional decomposition into an EOF basis would be meaningless (and highly sensitive to small changes in the data).

As previously stated, this is essentially a generalized Fourier approach. In fact, it was shown earlier (§D15.1) that sines and cosines (at harmonic frequencies) are *natural* eigenvectors of systems undergoing translation of the origin and of time-invariant linear systems. Thus, we would decompose our data into a set of orthogonal sines and cosines (the eigenvectors, $\mathbf{E}^{\mathrm{T}}$, in (15.18b)), and the variance would be spread over all frequencies, with the dominant variance captured by those harmonics that are closest to any true frequencies in the data. On the other hand, in principal components, we have found a different eigenvector basis, the one that is customized to the particular data set so that it captures the maximum variance in the fewest terms (which can be sines and cosines).

Finally, regarding the roots of the characteristic equation. If some of the eigenvalues are zero, it is an indication that the original matrix from which the eigenvalues are determined is singular. The true rank (number of independent data series) of the matrix is equal to the number of nonzero roots. For covariance matrices such as we are dealing with here, this should rarely if ever occur.

You may want to standardize the various data vectors going into the **A** matrix before you compute the EOFs. This leads to a correlation matrix instead of covariance matrix. For the covariance matrix, the data with the largest absolute variance may simply dominate the entire decomposition, since the other vectors may not contribute much to the overall variance. For the correlation matrix, all data vectors contribute more equally to the overall variance. The latter is important if one wants to find the relative contributions of each data set. The former is important if the decomposition of the total variance is what is desired.

If there are multiple roots of the same values, the matrix is called **defective**, and the resulting eigenvectors will not be distinct. This is fairly uncommon, as it indicates that the variance of two different variables is identical. It is also uncommon computationally due to round-off errors Fortunately, in symmetric matrices, this does not hurt us specifically, since the eigenvectors are still orthogonal, but in either case, the easiest solution is simply to perturb one of the numbers contributing to the multiple eigenvalue slightly, since the associated eigenvectors will most likely need to be ignored, given the fact that the uncertainties in the eigenvalues scale with the separation of the roots, and in this case the roots will be so close as to render the use of the associated eigenvectors meaningless. The number of equal eigenvalues is called the degree of **multiplicity**.

Uncertainties in eigenvalues are generally treated via a rule of thumb from North et al. (1982):

$$\Delta\lambda = \lambda(2/n)^{1/2}. \tag{15.49}$$

If $\Delta\lambda \geq \lambda_\ell - \lambda_{\ell-1}$, the two eigenvalues are too close together to separate. That is, uncertainty in one overlaps the value of its neighbor, so they are not distinct, and should be combined into a single mode.

Another related technique, **factor analysis**, differs from principal components in that while the latter is mainly interested in finding the fewest components that describe the most variance, the former is interested in finding those components that show the strongest degrees of correlation between the variables of the dataset. For example, it might collapse a set of 50 variables in 50 new components for which each component represents a natural clustering (as determined by correlation) of some subset of variables.

Figure 15.4 summarizes the above operations and results for the original Antarctic example presented in the previous figures.

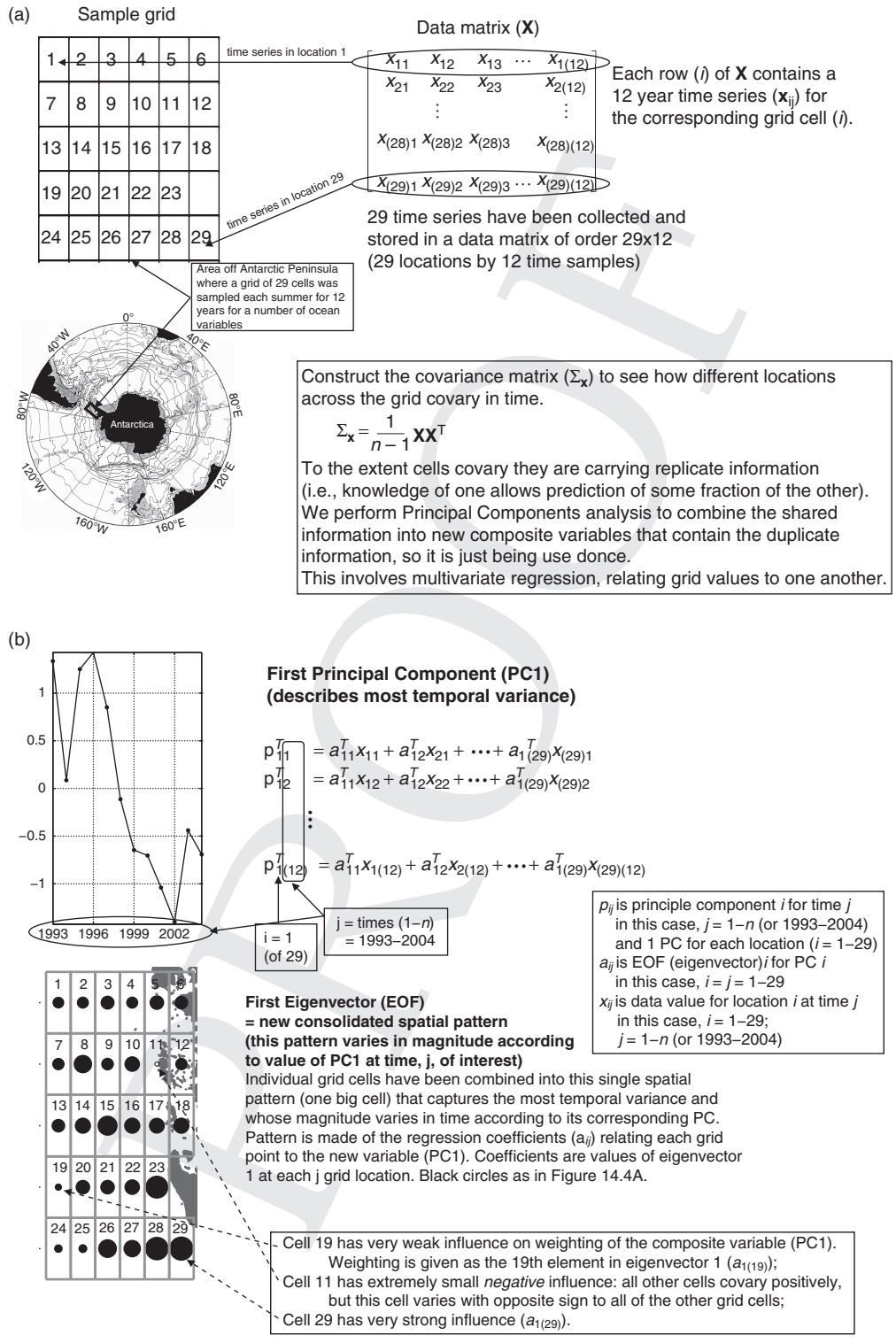


Figure 15.4 Overview of construction and interpretation of EOF analysis. (**A)** Data set of anomalies of a measure of upper-ocean salinity relative to the average (climatology), showing the sample grid and corresponding data matrix; (**B)** showing the first EOF and PC explaining the most variance;

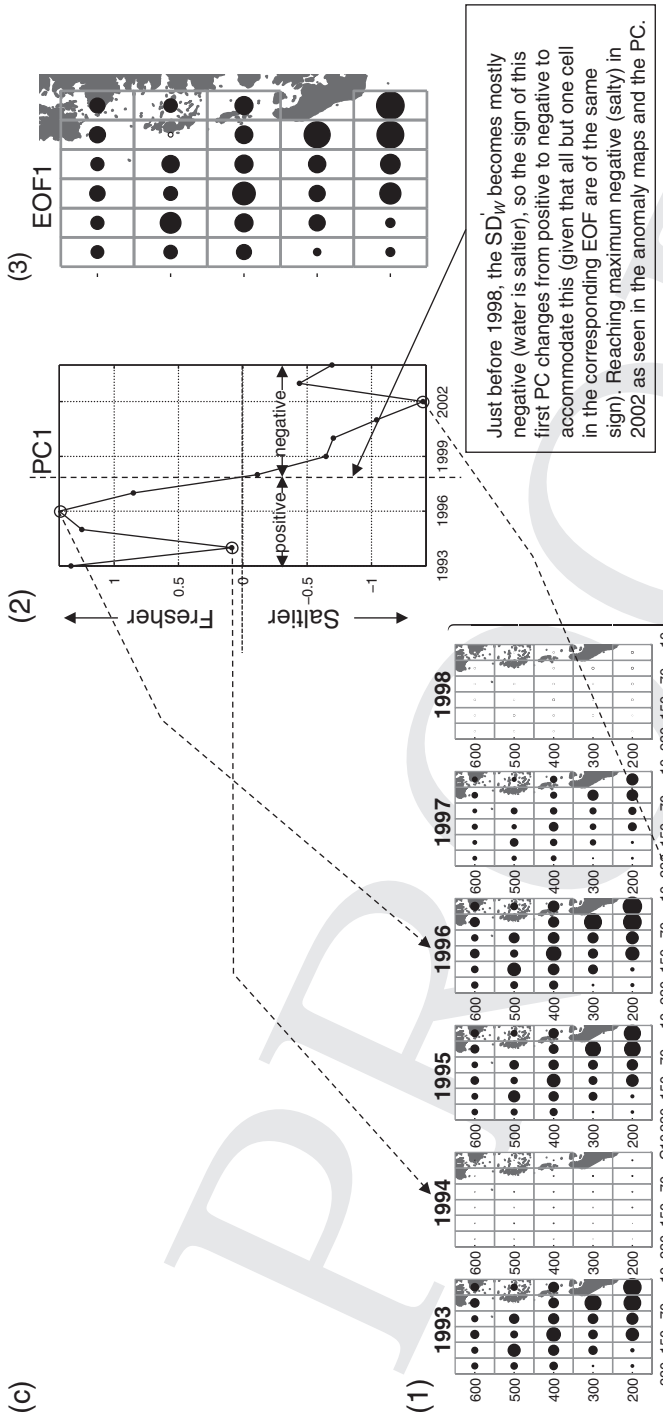


Figure 15.4 (Cont.) (**C**) complete description of the parts for the first PC and EOF pattern, particularly showing that the EOF pattern is multiplied by the value of the PC to reconstruct each particular year for this mode (note that when the PC crosses 0, the open circles in EOF1 become solid and vice versa for the solid circles);

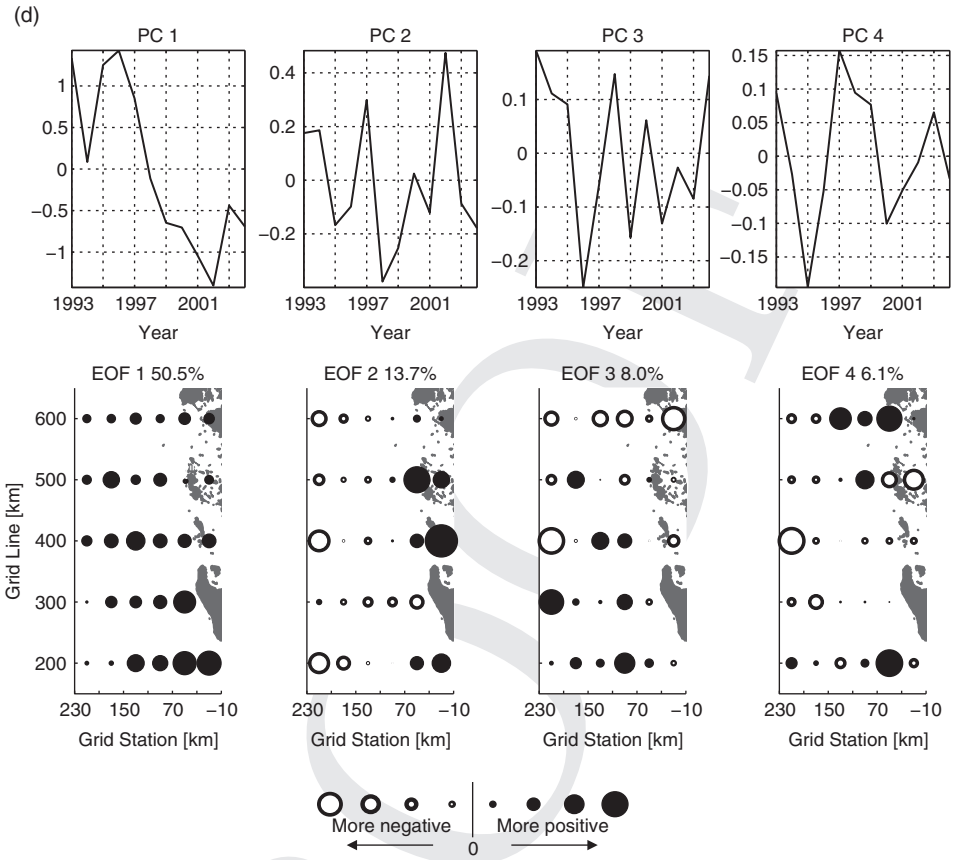


Figure 15.4 (cont.) **(D)** the first four principal components and eigenvectors and their percentage of the total variance described. This is a beautiful system: the *i*th eigenvector (EOF i) defines the pattern for that mode via the eigenvector elements, a_{ij} (the size of the circles in the figure are simply the a_{ij} values), and those elements are also the weights for the PC. Wow, we set up a potentially complicated constrained optimization problem and end up with an eigen-problem where everything is given by the eigenvectors!

15.5 Singular Spectrum Analysis (SSA)

Singular Spectrum Analysis (**SSA**; or "Singular Systems Analysis," in some circles) deals with the application of EOF analysis to single time series through a transform known as the Karhunen–Loéve transform. Its strength, as with EOF analysis in general, is that it allows you to find modes that are data adaptive (i.e., that fit the shape and internal consistency of the time series). Vautard et al. (1992; hereafter **VYG**) have provided an excellent review/survey of this topic from the standpoint of extracting signal (stochastic and deterministic) from noise in a data series. In keeping with our earlier attempt to decompose our data into a signal (or here, *intrinsic*) component and noise (*extrinsic*) component, VYG show how this is accomplished via SSA. Specifically, the lowest-order modes will represent the signal component, and how to determine this

number of modes is discussed at length in VYG (though that concentrates more heavily on chaos theory than on what is relevant here).

Trajectory Matrix

A fundamental difference from EOF analysis is the construction of the sample covariance matrix to be decomposed for the eigenstructure. Dynamic system analysis suggests that we can segment the data into m offset lagged segments. As long as the time series is weakly stationary (mean and variance, or in this case, autocovariance, which actually implies it is ergodic), each segment will contain the same relationship between lagged values, and it will be this lagged covariance that the method will maximize in the PCs.

For the 2-dimensional fields used with EOF analysis, you had a data matrix, with each row containing a time series with as many rows as stations, or grid points, sampled (n, the number of EOF modes). For a single time series, that matrix is a row vector. Constructing the sample covariance matrix for this in the usual manner would clearly be inadequate. Instead, we form a **trajectory matrix** whereby we examine the time series as offset (lagged) copies of itself. To construct this, you must choose a "window length," or **embedding dimension**, m, equivalent to the number of spatial locations in the EOF example (i.e., the size of the maps and number of eigenvectors).

Assuming stationarity and ergodicity, you can produce a much better estimate of the covariance matrix through the autocovariance serial product. This provides a reasonable estimate through at least some lags. This resolution is dictated by the embedding dimension m, which determines the number of modes that will be constructed to fit the data. It also sets the duration in time of the window over which the time series will be examined, so if you have an idea of the time scale of the signal you seek, then m should be chosen to resolve this (this will become more obvious below). Figure 15.5 demonstrates the construction of a trajectory matrix.

For standard spatiotemporal data, each *column* in the data matrix represented a "map" for a single time (i.e., the value of the variable at a single time for each station location). Then the eigenvectors give new composite variables showing spatial patterns that contain the most shared variance, with the PCs showing how that new pattern changes in time. For SSA, the trajectory contains m rows, equivalent to m spatial locations, but in this case the "maps" are segments in time over which the new composite variables will contain the most shared variance, with the PCs showing how those segments vary in time over the full extent of the time series.

After choosing m, remove the last $m - 1$ points from the times series, giving an effective length $n = N - m + 1$, and then make an $m \times n$ matrix, where each row is an n-length segment of the time series, but each row shifts this series, beginning and ending one data point later. So, for a time series of $N = 20$ points that runs from 1 to 20, we choose for an example an embedding dimension of $m = 5$ and form our $m \times n$ data matrix as shown in the example of (Figure 15.5).

(a) For a series of length N:

$x_t = x_1\, x_2\, x_3 \ldots x_N$

for an embedding dimension m, construct m single-lag offset segments (length $n = N–m+1$):

$$x_{t1} = x_1\, x_2\, x_3 \ldots x_n$$
$$x_{t2} = x_2\, x_3\, x_4 \ldots x_{n+1}$$
$$x_{t3} = x_3\, x_4\, x_5 \ldots x_{n+2}$$
$$\vdots$$
$$x_{tm} = x_{69}\, x_{70}\, x_{71} \cdots x_{n+m-1(=N)}$$

$$X_M = \underbrace{\begin{bmatrix} x_1 & x_2 & x_3 & \ldots & x_{n+0} \\ x_2 & x_3 & & & x_{n+1} \\ x_3 & & & & x_{n+2} \\ \vdots & & & & \vdots \\ x_m & & & & x_{n+(m-1)} \end{bmatrix}}_{N-m+1 \text{ length}} \Big\} m \text{ offset segments}$$

$$\Sigma_{mm} = \mathbf{X}_{mn}\mathbf{X}_{nm}{}^T = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12} & \ldots & \sigma_{1m} \\ & \sigma_{22}^2 & & \\ \vdots & & & \vdots \\ \sigma_{m1} & & & \sigma_{mm}^2 \end{bmatrix}$$

(b) For a series of length $N = 20$:

$x_t = x_1\, x_2\, x_3 \ldots x_{17}\, x_{18}\, x_{19}\, x_{20}$

An embedding dimension $m = 5$, construct m single lag offset segments of length $n = N – m + 1$:

$$\begin{array}{cccccccccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\ x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\ x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} \\ x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} \\ x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} & x_{20} \end{array}$$

$$X_M = \underbrace{\begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_{(N-m)+1} \\ x_2 & x_3 & & & x_{(N-m)+2} \\ x_3 & & & & x_{(N-m)+3} \\ \vdots & & & & \vdots \\ x_m & & & & x_{(N-m)+m} \end{bmatrix}}_{n\,(=N-m+1)\text{ length}} \Big\} m \text{ offset segments}$$

Trajectory matrix (X_M) is often defined as the transpose of what is shown here, which we prefer for consistency with our previous order of operations, so:

$\Sigma_X = XX^T$, not X^TX

$$\Sigma_{mm} = \mathbf{X}_{mn}\mathbf{X}_{nm}^T = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12} & \ldots & \sigma_{1m} \\ & \sigma_{22}^2 & & \\ \vdots & & & \vdots \\ \sigma_{m1} & & & \sigma_{mm}^2 \end{bmatrix}$$

True covariance matrix is a **Toeplitz** matrix, showing the following symmetry:

$$\Sigma_{mm} = \begin{bmatrix} \sigma_X^2 & \gamma_1 & \ldots & \gamma_m \\ \gamma_1 & \sigma_X^2 & \gamma_1 & \\ \vdots & \gamma_1 & & \vdots \\ & & & \gamma_1 \\ \gamma_m & & \gamma_1 & \sigma_X^2 \end{bmatrix}$$

Figure 15.5 (**A)** Generic example of trajectory matrix ($\mathbf{X}_M$) construction and of covariance matrix for diagonalization. (**B)** Explicit example of trajectory matrix for $N = 20$ and $m = 5$. (**C)** Showing how embedding dimension relates to number of spatial locations (forming maps at particular times) in standard EOF analysis of spatio-temporal data, as described earlier.

(c)

For a series of length $N = 20$:

$x_t = x_1\ x_2\ x_3\ \ldots\ x_{17}\ x_{18}\ x_{19}\ x_{20}$

An embedding dimension $m = 5$, construct m single lag offset segments of length $n = N - m + 1$:

$$\begin{array}{ccccccccccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\
x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\
x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} \\
x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} \\
x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} & x_{20}
\end{array}$$

Eigenvectors describe shape over segment

PC gives amplitude of shape for each segment (i.e., captures local features)

Each column of length m is effectively a "map", as was the case for the standard EOFs for our spatio-temporal example, but in this case each column represents a different segment of the full length time series from the first to last (N) point. So the eigenvectors will be patterns of length m, but in this case, not patterns in space, but rather patterns in time of length m. The PCs give how the amplitude of those patterns vary with the segment of time they represent.

Figure 15.5 (cont.)

$\mathbf{X}\mathbf{X}^T/n$ gives an $m \times m$ sample autocovariance matrix ($\hat{\Sigma}_X$), for which each diagonal, away from the principal diagonal, gives the lagged autocovariance, with the lag equal to the number of diagonals away from the principle diagonal.

You now diagonalize the covariance matrix as with EOFs, and then reconstruct the time series with any of the desired modes. Specifically,

$$\begin{aligned} \mathbf{E}^{\mathrm{T}}\Sigma_X\mathbf{E} &= \mathbf{\Lambda} \\ \mathbf{E}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{E} &= \mathbf{\Lambda} \end{aligned} \tag{15.50a}$$

define

$$\begin{aligned} \mathbf{P}_{mn} &= \mathbf{E}^{\mathrm{T}}_{mm}\mathbf{X}_{mn} \\ \mathbf{X}_{mn} &= \mathbf{E}_{mm}\mathbf{P}_{mn}. \end{aligned} \tag{15.50b}$$

With SSA, you now have another difference from EOFs. If you simply project the eigenvectors on the PCs, you will reconstruct the time series of the trajectory matrix (i.e., missing m points and at the various different lags). In order to reconstruct the entire time series, VYG use least squares to fit the modes to the entire data set, allowing reconstruction of the entire time series using any set of modes, $\mathrm{R}_{\mathfrak{Z}}$ (e.g., if you wish to reconstruct the time series using only the signal modes, say, 1 through 4, then $\mathrm{R}_{\mathfrak{Z}}$ would be the first three eigenvectors):

$$
\begin{aligned}
(R_{\Im}X)_i &= \frac{1}{i}\sum_{k\in\Im}\sum_{j=1}^{i} p_{i-j+1}^{k}E_j^k && 1 \le i \le m-1 \\
(R_{\Im}X)_i &= \frac{1}{m}\sum_{k\in\Im}\sum_{j=1}^{m} p_{i-j+1}^{k}E_j^k && 1 \le i \le N-m+1. \\
(R_{\Im}X)_i &= N-i+1\sum_{k\in\Im}\sum_{j=1-N+m}^{m} p_{i-j+1}^{k}E_j^k && N-m+2 \le i \le N
\end{aligned}
\tag{15.51}
$$

Picking Embedding Dimension

Picking the embedding dimension can proceed in a number of ways. EOFs provide for 2-dimensional data set patterns (EOFs) that share a common time history (PCs). Time one is showing the pattern of the entire spatial set (a map) at the first time (i.e., the first column of the data matrix), then the map at time two, etc. For SSA, the column length is dictated by m, so m serves to set the distance along the time series that will be fit as a pattern that is repeated in time with different amplitudes according to the PC. So, m should be chosen that is long enough to capture any changing features you wish to capture. Alternatively, you might simply use window closing – that is, choose an m that is likely too long, and one that is too short, and then some intermediate values, until you have a value where small changes in m yield similar modes.

We now apply this methodology to the LR04 record introduced previously in the text in Figure 15.6. First, this method should be ideal for identifying the stepwise trend displayed by the data, so perhaps after removing this trend we will be able to more definitively answer the original example question as to whether long-term climate shows two modes (cold and warmer periods) or some sort of smoother continuum of climate. This curve contains $N = 5321$ data points, and we use m =100. The eigenvalues are presented in a scree plot (Figure 15.7) to gauge which ones dominate the variance. Clearly, the leading mode (mode 1) dominates, as one might expect from simple visual inspection of LR04. Modes 2 and 3 and modes 4 and 5 fail North's test (15.49) for distinct modes, and thus are combined into coupled reconstructed modes. Such pairs often represent regular oscillations in the data, the nonlinear equivalent of a Fourier sine-cosine pair.

When first introducing LR04 in Chapter 3 (example of PMF), we raised the question of whether climate was bimodal with two stable states: glacial (ice age) and interglacial (ice free). Our initial examination showed a hint of bimodality, but that was later seen to be a consequence of the warmer climate prior to 1 million years ago. We then investigated means of removing the cooling trend (climate cooling from 3 to 1 million years ago) by fitting various curves to LR04 to remove this long-term trend. With better fits to the curve, we starting seeing a loss of bimodality toward a more symmetrical distribution of climate. Here we have now found the best fit of the trend through the first SSA mode. When we remove this mode and reconstruct only the signal modes (two and three, and

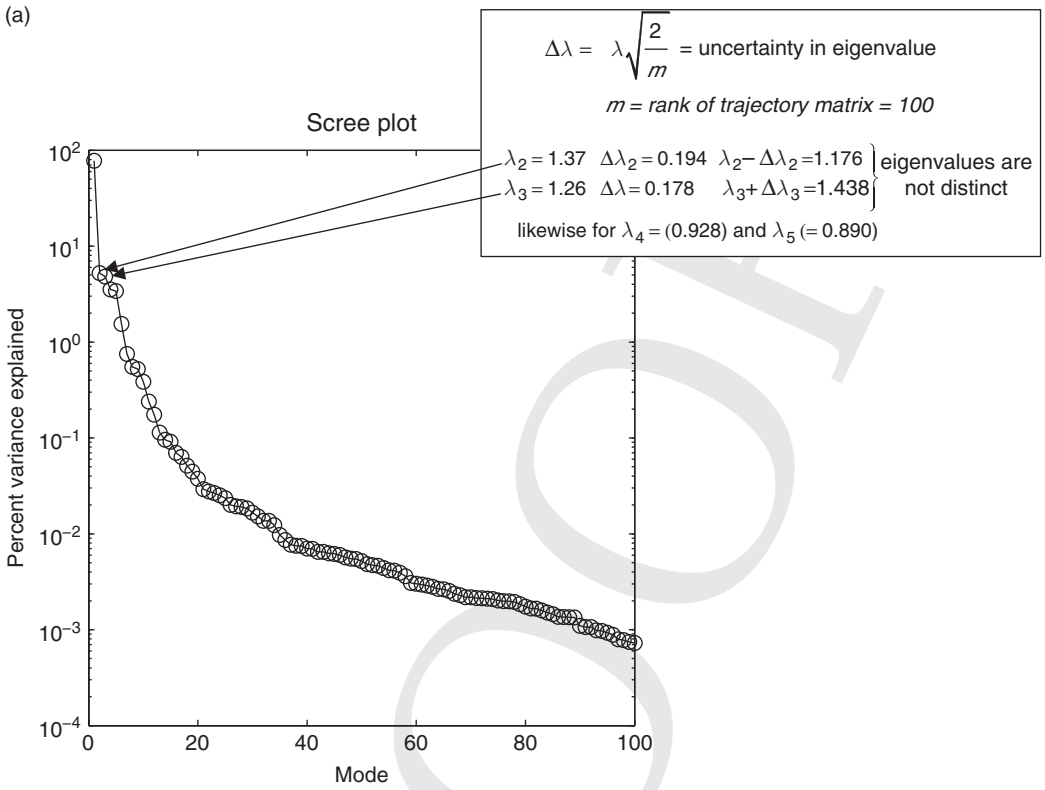


Figure 15.6 (**A)** Scree plot for LRO4 with $N = 5321$ and $m = 100$. Using North's rule of thumb, it is shown that eigenvalues two and three as well as four and five are too close together to separate, so those pairs are combined into a single reconstructed series. (**B)** First three reconstructed series. The first mode clearly captures the trend (describing 78 percent of the total LR04 variance), while the second series (modes two and three) appears to capture the "eccentricity" signal (highly variable, but centered on ~100 kyr cycle in the most recent million years) of the ice ages, describing ~10 percent of the variance (it is clearly more dominant in the most recent million years). The third series (modes four and five) captures the "obliquity" of Earth's axis (a ~41 kyr cycle) that captures only 7 percent of the variance (also, some of that signal is captured in the second series).

four and five) our PMF for this reconstructed signal version of LR04 show a nearly perfect Gaussian distribution (Figure 15.7).

15.6 Take-Home Points

1. EOF analysis is a means of reducing a large dependent dataset into a new set of variables that contain the information in a minimum of orthogonal variables, reducing all dependent information into these few new variables that are completely independent.
2. This information can be used for optimal interpolation, whereby if a data point is missing, the EOFs know how that point covaried with every other point in the grid, and that information can be used to estimate the most likely value of the missing datum.

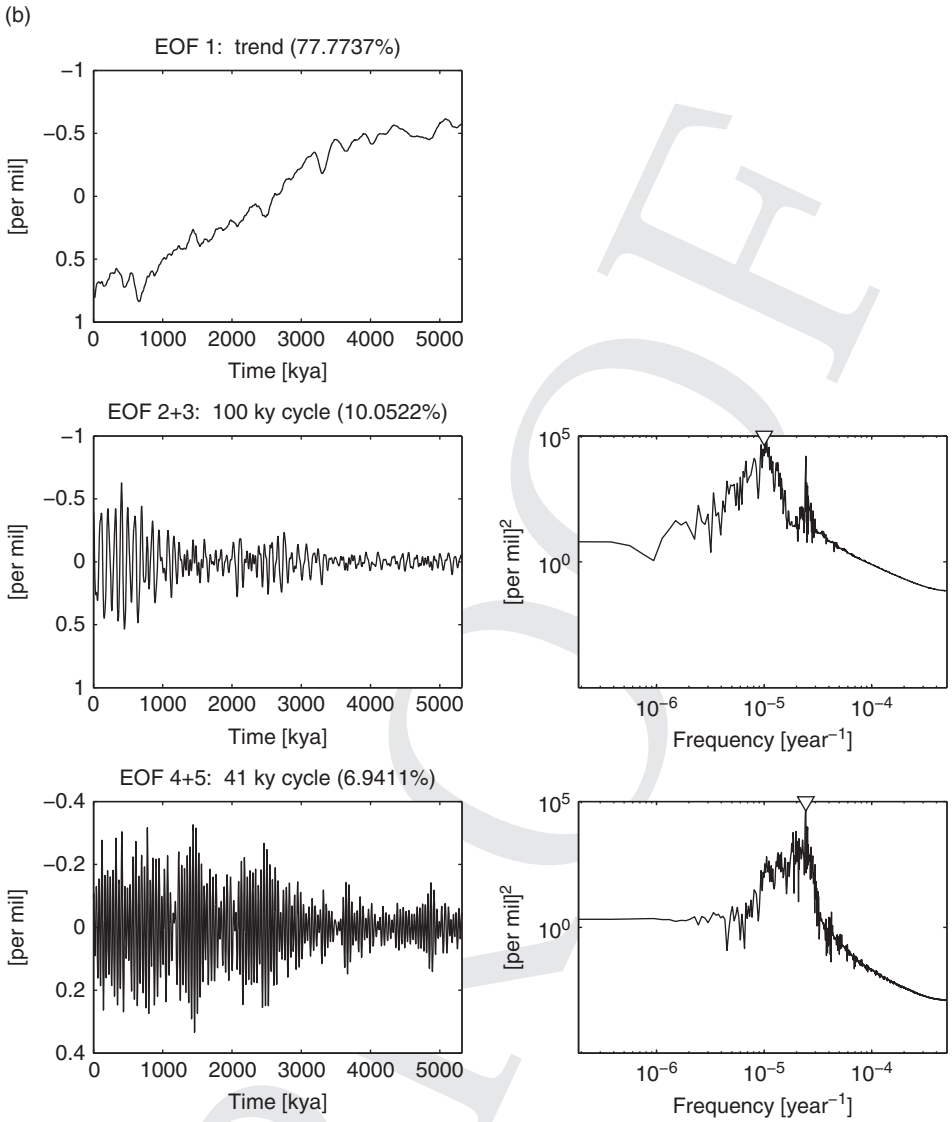


Figure 15.6 (cont.)

3. By decomposing a data set into orthogonal functions with no covariability, one hopes that some of the empirical orthogonal functions (the EOFs) explain the underlying processes responsible for the random process (the times dataset).
4. An EOF decomposition of a time series (called Singular Spectrum Analysis; SSA), is functionally like a Fourier transform, but the orthogonal functions fitting the series are not harmonics or known mathematical functions; rather, they are functions that best fit the shape and internal consistency of the time series. This is a data-adaptive decomposition and is ideal for finding signal in noise.

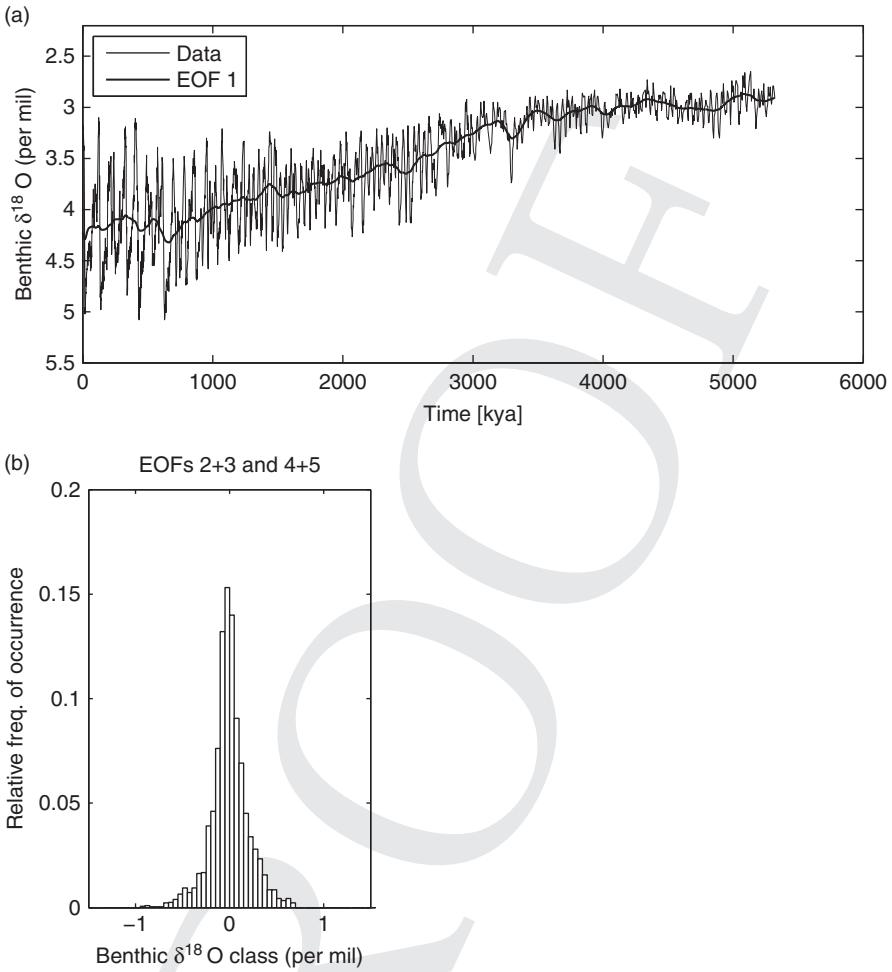


Figure 15.7 (**A)** Showing the fit of the first EOF to original data. It clearly captures the trend, as well as a few bumps here and there. (**B)** Probability mass function (PMF) showing distribution of glacial (cold; negative) and interglacial (warm; positive) periods of the series reconstructed from "signal modes" (two and three, and four and five), suggesting that natural climate variability is evenly distributed about a variable-mean state over the last 5 million years.

15.7 Questions

Pencil and Paper Questions

1. a. Regarding EOF analysis:

    1. What does EOF mean?
    2. Describe your data matrix for such an analysis.

    b. Show the equations needed to do the EOF analysis, from manipulating the data matrix to making the PCs (don't need derivations; instead look for the fundamental matrix operations leading to and resulting in producing PCs).

2. a. Regarding SSA analysis:

    1. What does SSA mean?
    2. Describe your data matrix for such an analysis.

   b. Show the equations needed to do SSA, from manipulating the data matrix (give its name after describing) to making the PCs (don't need derivations; instead look for the fundamental matrix operations leading to and resulting in producing PCs).

Computer-Based Questions

1. Using LR04 and GISP2 truncated to an equal length in time, remove a nonstationary mean via SSA, then compute PSD for the first modes of each, and compare and contrast differences.

   Johnson and Wichern (1988) have an excellent discussion on the geometric interpretation of PCs (pages 49–50) and on spectral decomposition, just before that.

Appendix 1 Overview of Matrix Algebra

A1.1 Overview

Matrix algebra (also called *linear algebra*) is the branch of mathematics that deals with the algebraic manipulation of groups of numbers or equations stored in matrices. The advantage of matrix algebra lies in the fact that it provides concise and simple methods for manipulating large sets of numbers such as those frequently encountered in data analysis. As such, it is ideal for computers, and the compact form of matrices allows convenient notation for describing large tables of data. Matrix operations allow you to see complex relationships that would otherwise be obscured by the sheer size of the data (i.e., it aids clarification); most matrix manipulation involves just a few standard operations for which standard subroutines are readily available.

This review mainly introduces the terminology and tools available in matrix algebra. These will then be applied in numerous analysis techniques.

A1.2 Definitions

A **matrix** is a rectangular array of "elements" arranged in a series of m rows and n columns (like a table). A matrix is most commonly indicated by a boldface letter (e.g. **A, B, x**), an uppercase letter, or a letter with an underscore (e.g., $\underline{a}$ or $\tilde{a}$), in brackets (e.g., $[a]$, $[A]$), or with hat (e.g. $\hat{a}$). Here, matrices will consistently be indicated by a boldface letter.

Order of a matrix is the specification of the number of rows by the number of columns. Order for a matrix **A** is usually given as $m \times n$ (read, "m-by-n") or $\mathbf{A}_{m,n}$. Typically, the letters m, n and p are used to indicate order (row position is always given first).

Elements of a matrix **A** are given as a_{ij}, where the value of i specifies the row position and the value of j specifies the column position (the letters k and l are often used in place of i and j as well). An element can be a number (real or complex), algebraic expression or (with some restrictions) a matrix or matrix expression, or other component.

As a convention here (allowing all matrix manipulations to be presented in a common form), data matrices (i.e., matrices in which the elements represent data values) are constructed so that the rows of any one column contain the observations related to a particular variable or location. For example, the first column may contain all of the temperature observations, the second column the salinities, etc.; or the first column may contain the temperatures at location 1, the second column the temperatures from location 2, etc. If the data represent time or space series, typically the rows correspond to time or space. Since there are typically more observations than variables, such data matrices are usually rectangular, having more rows (m) than columns (n).

Column vector is a matrix containing only a single column of elements:

$$\tilde{a}_n = A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix}$$

Figure A1.1 Example of column vector containing n elements (dimension n).

Box A1.1 Example of Matrix Contents and Labeling

$$\begin{array}{r} \\ \text{A} := \text{m rows} \end{array} \begin{array}{c} \text{– n columns –} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \end{array} = \begin{array}{c} \text{j = 1} \quad \text{j = 2} \quad \text{j = 3} \\ \begin{bmatrix} 12 & 4 & 10 \\ 8 & 1 & 11 \\ 15 & 3 & 7 \\ 14 & 1 & 9 \end{bmatrix} \end{array} \begin{array}{l} \\ \text{i = 1} \\ \text{i = 2} \\ \text{i = 3} \\ \text{i = 4} \end{array}$$

This matrix, **A**, has order $m \times n = 4 \times 3$ (4 rows by 3 columns), the element $a_{23} = 11$, $a_{13} = 10$, etc.

Row vector is a matrix containing only a single row of elements:

$$\widetilde{a}_n = A = [a_1\ a_2\ a_3 \fallingdotseq a_n].$$

Vector size is simply the number of elements it contains (= n, in both examples above).

Note that some people do not explicitly differentiate between row and column vectors. Instead, they simply assume that the "vector" takes the form (row or column) required for the operation in which it is being used.

Summing vector, written as $\mathbf{1}_n$, is a vector of order n, in which all n elements are equal to the value 1. This vector is useful when the sum of a vector's elements is required.

Null matrix, written as $\mathbf{0}$ or $\mathbf{0}_{(m,n)}$, has all elements equal to 0. It plays the role of zero in matrix algebra.

Square matrix has the same numbers of rows as columns, so order is $n \times n$.

Diagonal matrix is a *square matrix* with zero in all positions except along the **principal diagonal** (or **lead diagonal**) as shown in (A1.1):

$$\mathbf{D} = \begin{bmatrix} 3 & & \\ & 5 & \\ & & 5 \end{bmatrix} \tag{A1.1a}$$

or

$$\mathbf{A} = \mathbf{A}^{\mathrm{T}} = \begin{bmatrix} 1 & 2 & 5 \\ 2 & 7 & 3 \\ 5 & 3 & 9 \end{bmatrix} = \text{symmetric.} \tag{A1.1b}$$

Those elements that do not lie on the principal diagonal are often called **off-diagonal** or nondiagonal elements.

Diagonal matrices are important for scaling rows or columns of other matrices. Note that some authors do not require that a diagonal matrix be square – they only require that the a_{ii} be nonzero and all other elements be zero.

Identity matrix (**I**) is a *diagonal matrix* with all of the nonzero elements – that is, the diagonal elements (the a_{ii}), equal to 1. Written as $\mathbf{I}$ or $\mathbf{I}_n$, it plays the role of 1 in matrix algebra. So,

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A}. \tag{A1.2}$$

The order of $\mathbf{I}$ will always be taken to be conformable (this is discussed below) with that of the matrix it is multiplying.

Unit matrix is one in which *all* elements are equal to 1. This matrix is useful in some operations.

Lower-triangular matrix is a *square matrix* with all elements equal to zero *above* the principal diagonal:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 7 & 0 \\ 2 & 3 & 9 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 3 & 7 & \\ 2 & 3 & 9 \end{bmatrix} \tag{A1.3a}$$

or

$$a_{ij} = \begin{cases} 0 & i < j \\ \text{nonzero} & i \geq j. \end{cases} \tag{A1.3b}$$

Upper-triangular matrix is a *square matrix* with all elements equal to zero *below* the principal diagonal:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 5 \\ 0 & 7 & 3 \\ 0 & 0 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 5 \\ & 7 & 3 \\ & & 9 \end{bmatrix} \tag{A1.4a}$$

or

$$a_{ij} = \begin{cases} 0 & i > j \\ \text{nonzero} & i, \leq j. \end{cases} \tag{A1.4b}$$

If one multiplies two triangular matrices of the same form, the result is a third matrix of the same form.

A **fully populated matrix** is a matrix with all of its elements nonzero.

A **sparse matrix** is a matrix with only a small proportion of its elements nonzero. From a computational standpoint, this type of matrix is often treated using techniques specially formulated to take advantage of the relatively few elements contained in the matrix.

A **scalar** is a number (or a matrix with a single element).

Matrix transpose (or transpose of a matrix) is obtained by interchanging the rows and columns of a matrix. So row i becomes column i and column j becomes row j; also, the order of the matrix is reversed from $m \times n$ to $n \times m$. The transpose of matrix $\mathbf{A}$ is denoted $\mathbf{A}^{\mathrm{T}}$, or sometimes $\mathbf{A}'$:

$$\mathrm{A} = \begin{bmatrix} 1 & 2 & 5 \\ 6 & 7 & 3 \\ 4 & 1 & 9 \end{bmatrix}; \mathbf{A}^{\mathrm{T}} = \begin{bmatrix} 1 & 6 & 4 \\ 2 & 7 & 1 \\ 5 & 3 & 9 \end{bmatrix}$$

So,

$$a_{ij} \rightarrow a_{ji}. \tag{A1.5b}$$

A diagonal matrix is its own transpose: $\mathbf{D}^{\mathrm{T}} = \mathbf{D}$.

Symmetric matrix is a *square matrix* that is symmetrical about its principal diagonal, so

$$a_{ij} = a_{ji}. \tag{A1.6a}$$

A symmetrical matrix, $\mathbf{A}$, is equal to its own transpose:

$$\mathbf{A} = \mathbf{A}^{\mathrm{T}} = \begin{bmatrix} 1 & 2 & 5 \\ 2 & 7 & 3 \\ 5 & 3 & 9 \end{bmatrix} = \text{symmetric}. \tag{A1.6b}$$

Skew symmetric matrix is a *square matrix* in which

$$a_{ij} = -a_{ij}, \tag{A1.7a}$$

so

$$\mathbf{A}^{\mathrm{T}} = -\mathbf{A} \tag{A1.7b}$$

and

$$a_{ii} = 0.\ 1 \tag{A1.7c}$$

That is, the principal diagonal elements are zero:

$$\mathbf{A} = \begin{bmatrix} 0 & -2 & 5 \\ 2 & 0 & 3 \\ -5 & -3 & 0 \end{bmatrix} = \text{skew symmetric.} \tag{A1.7d}$$

Any square matrix can be split into the sum and difference of a symmetric and skew symmetric matrix:

$$\mathbf{A} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^{\mathrm{T}}) + \frac{-1}{2}(\mathbf{A} - \mathbf{A}^{\mathrm{T}}). \tag{A1.8}$$

Though it looks a bit goofy, this operation is extremely important throughout matrix algebra.

A1.3 Basic Matrix Operations

Addition and subtraction require matrices of the same order, since this operation simply involves addition or subtraction of corresponding elements. This is shown in Figure A1.2.

Also,

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \tag{A1.9}$$

$$(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C}). \tag{A1.10}$$

Multiplication operations fall into several groups.

Scalar multiplication of a matrix involves multiplying a matrix by a constant (scalar). For a scalar β,

$$\beta\mathbf{A} = \beta \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} \beta a_{11} & \beta a_{12} & \beta a_{13} \\ \beta a_{21} & \beta a_{22} & \beta a_{23} \\ \beta a_{31} & \beta a_{32} & \beta a_{33} \end{bmatrix}, \tag{A1.11}$$

$$\mathbf{A} + \mathbf{B} = \mathbf{C}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}; \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} \end{bmatrix}$$

Figure A1.2 Example of adding two matrices together.

so the product is a matrix in which each element has been multiplied by β.

Scalar product (also **dot product** or **inner product**) is the (scalar) product of two *vectors* of the same size, defined as

$$\mathbf{A} \cdot \mathbf{B} = \beta, \tag{A1.12a}$$

where

A = a row vector (or the transpose of a column vector) of length *n*
B = a column vector (or the transpose of a row vector), also of length *n*
β = the *scalar* product of **A**·**B**.

Specifically,

$$\mathbf{A} = [a_1 \quad a_2 \quad a_3]; \quad \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{A1.12b}$$

and

$$\beta = a_1b_1 + a_2b_2 + a_3b_3, \tag{A1.12c}$$

so the product is *not* a matrix, but rather a scalar (i.e., a number).

Some people like to visualize this multiplication as shown in Figure A1.3, where the corresponding elements are multiplied (product indicated along the diagonal lines) and added to make the scalar product.

Conceptually, this product can be thought of as multiplying the length of one vector by the component of the other vector that is parallel to the first.

In terms of force vectors, |**B**|cosθ represents the component of force **B** acting to drive a displacement **A** in the direction indicated. The dot product gives the work in direction A. Alternatively, in terms of a decomposition, it gives that portion of the **B** vector that is represented along the **A** direction (the axis of one of the decomposing vectors). In this sense, the dot product is given as

$$\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}||\mathbf{B}|\cos\theta = \beta, \tag{A1.12c}$$

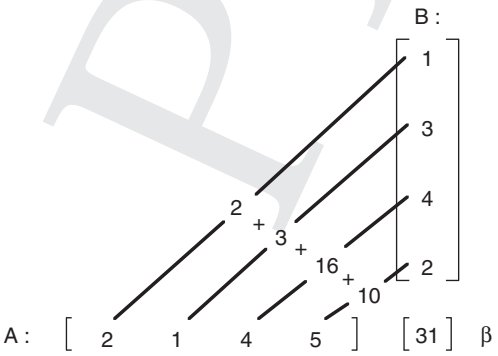


Figure A1.3 Schematic showing visualization of dot product, cell-by-cell multiplication and sum.

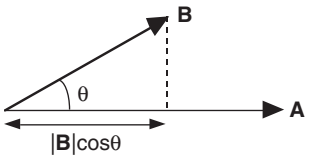


Figure A1.4 Dynamic schematic of dot product, where vector **B** is operating on vector **A**.

where |**x**| is the vector x **magnitude**, given as

$$|x| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots x_n^2}. \tag{A1.12d}$$

This is also referred to as a **projection** of **B** on **A**, which looks similar to a correlation (discussed later), and is given in summation form as

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} a_i b_i. \tag{A1.13}$$

Maximum principle says that the unit vector, **n** (a vector of magnitude unity), making **A**·**n** a maximum, is that unit vector pointing in the same direction as **A**. This is seen from the above definition of the scalar product, where, if

$$\mathbf{n} \parallel \mathbf{A}, \tag{A1.14a}$$

then

$$\cos\theta = \cos 0° = 1, \tag{A1.14b}$$

so

$$\mathbf{A} \cdot \mathbf{n} = |\mathbf{A}||\mathbf{n}|\cos\theta = |\mathbf{A}||\mathbf{n}| = |\mathbf{A}|. \tag{A1.14c}$$

For any other angle θ, the cos θ will be less than 1, yielding a smaller scalar product. This principle holds equally well for any vector **n** that is not a unit vector (note that **n** is usually used to denote a unit vector). So the vector that is parallel to **A** will give the largest dot product with **A**.

Parallel vectors (or **dependent vectors**) *geometrically* thus have cos θ = 1. So, if

$$\mathbf{A} \| \mathbf{B},$$

then

$$\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}||\mathbf{B}|. \tag{A1.14}$$

Also,

$$\mathbf{A} = \beta\mathbf{B}, \tag{A1.15a}$$

This latter formula states that if two vectors are parallel, they are related by a simple constant:

$$\beta = \frac{|\mathrm{A}|}{|\mathrm{B}|}. \tag{A1.15b}$$

The fact that parallel vectors are related by a constant indicates that they are **dependent** vectors. This represents an important concept in matrix algebra.

Orthogonal vectors are *perpendicular*, and thus have

$$\cos\theta = \cos 90° = 0,$$

so

$$\mathbf{A} \cdot \mathbf{B} = 0 \tag{A1.16}$$

when

$$\mathbf{A} \perp \mathbf{B}.$$

Orthogonal vectors represent a special limiting case of independent vectors. They have properties that make them ideal for a variety of mathematical operations.

Defining Independent, Orthogonal and Uncorrelated Vectors

For vectors, or time series: $\mathbf{X}_1$, $\mathbf{X}_2$, …, $\mathbf{X}_n$ in terms of vector operations, the following material holds.

Vectors are only dependent when they are parallel (one being shorter, longer or reversed) relative to each other so that multiplication by a constant can make them identical. If they are not parallel, they are *independent*, satisfying for any constants (except 0), a, b, …, z

$$a\mathbf{X}_1 + b\mathbf{X}_2 + \cdots + z\mathbf{X}_n \neq 0. \tag{A1.17a}$$

If they are at right angles to one another, as independent as can be, they are *orthogonal*, satisfying

$$\mathrm{X}_1^{\mathrm{T}}\mathrm{X}_2 = 0. \tag{A1.17b}$$

They are *uncorrelated* if

$$(\mathbf{X}_1 - \overline{\mathbf{X}_1})^T(\mathbf{X}_2 - \overline{\mathbf{X}_2}) = 0. \tag{A1.17c}$$

This says that the centered (i.e., vectors with means removed) are orthogonal, but that does not mean that the uncentered vectors are orthogonal (e.g., consider each vector as having a trend for a mean, which immediately forces a dependence).

Squaring vectors is simply

$$\text{For row vectors} : \mathbf{A}^2 = \mathbf{A}\mathbf{A}^{\mathrm{T}}$$

and

$$\text{For column vectors} : \mathbf{A}^2 = \mathbf{A}^{\mathrm{T}}\mathbf{A}. \tag{A1.18}$$

Matrix multiplication requires **conformable matrices**, which are those in which there are as many columns in the first as there are rows in the second, so

$$\mathbf{C}_{(m,n)} = \mathbf{A}_{(m,p)}\mathbf{B}_{(p,n)}, \tag{A1.19}$$

and the product matrix **C** is of order $m \times n$ with has elements c_{ij}. The elements c_{ij}, of **C**, are given by

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}. \tag{A1.18b}$$

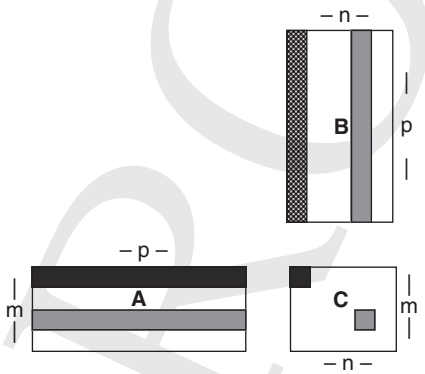
This is an extension of the scalar product – in this case, each element of **C** represents the scalar product of a row vector in **A** and column vector in **B**,

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

or

$$c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}.$$

In "box form," this product is shown by



This form graphically illustrates that the elements of **C** correspond to the scalar products of the vectors indicated (i.e., consider the matrices to be composed of row and column vectors). The element locations in **C** occur where the extensions of the vectors intersect in the **C** matrix. Note that the order is shown for each matrix – **C** will have as many columns as **B** has (n) and as many rows as **A** has (m).

Order of multiplication is important – usually.

$$\mathbf{AB} \neq \mathbf{BA}, \tag{A1.20}$$

unless **A** and **B** are square matrices of the same order. Otherwise, one of the two products cannot even be formed (since the matrices will not be conformable in both directions).

Order is specified by stating that

A is **pre-multiplied** by **B** (for **BA**); or
A is **post-multiplied** by **B** (for **AB**).

Multiple products can be multiplied as

$$\mathbf{D} = (\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}).$$

Computational Considerations

When examining a matrix multiplication, it is seen that the *order* in which pairs are multiplied in a multiple product can make a large computational difference. Consider the following example.

Box A1.2 Example of Matrix Indexing

$$\mathbf{C}_{(m,n)} = \mathbf{A}_{(m,p)}\mathbf{B}_{(p,n)}$$

This product involves $m \times n \times p$ multiplications ($m \times n$ vector products times p element multiplications in each product) and $m \times n \times (p - 1)$ additions (computed in a similar manner). If the matrix **E** is then the product,

$$\mathbf{E}_{(m,n)} = [\mathbf{A}_{(m,p)}\mathbf{B}_{(p,q)}]\mathbf{C}_{(q,n)},$$

the product in brackets involves mpq multiplications and the product of this with **C** then involves another mqn multiplication, for a total of mpq + mqn = mq(p + n) multiplications (and a nearly comparable number of additions).

Alternatively, if the product is multiplied as

$$\mathbf{E}_{(m,n)} = \mathbf{A}_{(m,p)}[\mathbf{B}_{(p,q)}\mathbf{C}_{(q,n)}],$$

the bracketed product involves pqn multiplications and the total product pqn + mpn = pn(q + m) multiplications.

Therefore,

1) $(\mathbf{AB})\mathbf{C} \rightarrow$ mq(p + n) multiplications;
2) $\mathbf{A}(\mathbf{BC}) \rightarrow$ pn(q + n) multiplications.

If **A** and **B** are both 100×100 order matrices and **C** is 100×1 order, then

m = 100
p = 100
q = 100
n = 1,

so the order of multiplication shown in 1) requires ~10^6 multiplications and 2) requires ~10^4 multiplications. Consequently, if one were to simply multiply **AB** and then this product by **C**, the computational cost would be nearly a million multiplications (and an almost equal number of additions) more than required by multiplying **BC** first and then pre-multiplying this product by **A**!

This is not just important for the obvious reason of computational time, but more importantly, the more the operations, the greater the accumulation of round-off error when dealing with very large matrices.

Transpose of a matrix product is simply multiplication by the transpose of the individual matrices in reverse order:

$$\mathbf{D} = \mathbf{ABC} \tag{A1.21a}$$

$$\mathrm{D}^{\mathrm{T}} = \mathrm{C}^{\mathrm{T}}\mathrm{B}^{\mathrm{T}}\mathrm{A}^{\mathrm{T}}. \tag{A1.21b}$$

A1.4 Special Matrix Products

Multiplication by I

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A} \tag{A1.22}$$

as seen by

$$\mathbf{A}\begin{bmatrix} 3 & 6 & 9 \\ 2 & 8 & 7 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{I} \quad .$$
$$\begin{bmatrix} 3 & 6 & 9 \\ 2 & 8 & 7 \end{bmatrix}\mathbf{AI}$$

Recall that the identity matrix, **I**, acts as 1 in matrix multiplications, and is always assumed to be conformable with the matrix being multiplied. Thus in (A1.22), if **A** has order $m \times n$, when post-multiplying **A** by **I, I** has order n (recall that **I** is a square matrix, so order n is equivalent to order n,n). When pre-multiplying **A** by **I, I** has order m.

Pre-multiplication by a diagonal matrix produces a matrix in which each *row* is scaled by a diagonal element of **D**. So, for

$$\mathbf{C} = \mathbf{DA}, \tag{A1.23}$$

where **D** is a diagonal matrix, and

$$\mathrm{D} = \begin{bmatrix} \mathrm{d}_{11} & & \\ & \mathrm{d}_{22} & \\ & & \mathrm{d}_{33} \end{bmatrix}; \quad \mathrm{A} = \begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & \mathrm{a}_{13} \\ \mathrm{a}_{21} & \mathrm{a}_{22} & \mathrm{a}_{23} \\ \mathrm{a}_{31} & \mathrm{a}_{23} & \mathrm{a}_{33} \end{bmatrix}$$
$$\mathrm{C} = \begin{bmatrix} \mathrm{a}_{11}\mathrm{d}_{11} & \mathrm{a}_{12}\mathrm{d}_{11} & \mathrm{a}_{13}\mathrm{d}_{11} \\ \mathrm{a}_{21}\mathrm{d}_{22} & \mathrm{a}_{22}\mathrm{d}_{22} & \mathrm{a}_{23}\mathrm{d}_{22} \\ \mathrm{a}_{31}\mathrm{d}_{33} & \mathrm{a}_{32}\mathrm{d}_{33} & \mathrm{a}_{33}\mathrm{d}_{33} \end{bmatrix}$$

so the first row of **C** is scaled by d_{11}, the second row by d_{22}, etc.

Post-multiplication by a diagonal matrix produces a matrix in which each *column* has been scaled by a diagonal element of **D**. So, for

$$\mathbf{C} = \mathbf{AD}. \tag{A1.24}$$

Then,

$$C = \begin{bmatrix} c_{11}d_{11} & c_{12}d_{11} & c_{13}d_{11} \\ c_{21}d_{22} & c_{22}d_{22} & c_{23}d_{22} \\ c_{31}d_{33} & c_{32}d_{33} & c_{33}d_{33} \end{bmatrix},$$

where the first column of **C** is scaled by d_{11}, the second by d_{22}, etc.

A1.4.1 Determinant of a Matrix

The determinant of a matrix is a single number representing a *property of a square matrix* (its exact meaning is dependent upon what the particular matrix represents). The main use here is to find the inverse of a matrix or solve simultaneous equations for which the determinant plays a major role. Symbolically, the determinant is usually given as det **A**, |**A**| or ||**A**|| (to differentiate it from magnitude).

Calculation of a 2 × 2 determinant is given by

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \quad \text{(A1.25)}$$

This is the difference of the cross products (the determinant of a 1×1 matrix is just the particular element).

Calculation of an *n*×*n* determinant is given by

$$|A| = a_{11}M_{11} - a_{12}M_{12} + a_{13}M_{13} - \ldots. - (-1)na_{1n}M_{1n}, \quad \text{(A1.26)}$$

where

M_{11} = the determinant, with the first row and column missing from the matrix;
M_{12} = the determinant, with the first row and second column missing; etc.

Generalizing, the determinant with the *i*th row and *j*th column removed is known as the **cofactor**, $(-1)i+jM_{ij}$. This will be a useful concept later for inverting a matrix.

This formula can be used recursively. That is, once the first column and row of the matrix have been eliminated (to construct M_{11}), then the determinant of the remaining portion of the matrix can be computed by applying the above formula to this reduced matrix so that it, too, can be reduced until a manageable size is attained (e.g., 3×3 or smaller). But note that, in any case, the cofactors are only those that correspond to the first row of elements in the matrix. That is, as seen in (A1.26), the cofactors are multiplied by a_{11}, a_{12}, etc., never by a_{21}, a_{nn}. However, the cofactors themselves can involve the determinants in which these other rows are included if a recursive implementation is required. That is, the cofactor for M_{11} might involve cofactors of submatrices whose determinants involve a_{22}, a_{23}, etc.

For a 4×4 determinant, each M_{ij} would be an entire expansion like that given above for the 3×3 determinant – one quickly needs a computer. That is, the M_{11} term would eliminate the first row and first column of the matrix, leaving a 3×3 matrix. This individual 3×3 matrix would then be expanded as in the above example, allowing

Box A1.3 Example of a Determinant

Example of a 3×3 determinant:

$$|A| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$M_{11} = \begin{vmatrix} \cancel{a_{11}} & \cancel{a_{12}} & \cancel{a_{13}} \\ \cancel{a_{21}} & a_{22} & a_{23} \\ \cancel{a_{31}} & a_{32} & a_{33} \end{vmatrix} = a_{22}a_{33} - a_{23}a_{32}$$

$$M_{12} = \begin{vmatrix} \cancel{a_{11}} & \cancel{a_{12}} & \cancel{a_{13}} \\ a_{21} & \cancel{a_{22}} & a_{23} \\ a_{31} & \cancel{a_{32}} & a_{33} \end{vmatrix} = a_{21}a_{33} - a_{23}a_{31}$$

$$M_{13} = \begin{vmatrix} \cancel{a_{11}} & \cancel{a_{12}} & \cancel{a_{13}} \\ a_{21} & a_{22} & \cancel{a_{23}} \\ a_{31} & a_{32} & \cancel{a_{33}} \end{vmatrix} = a_{21}a_{32} - a_{22}a_{31}$$

Applying the above formula gives

$$\begin{aligned} |A| &= a_{11}M_{11} - a_{12}M_{12} + a_{23}M_{13} \\ &= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}). \end{aligned}$$

solution for the M_{11} term. For the M_{12} term, the elimination of the first row and second column would also leave a 3×3 matrix, which would then be expanded to give M_{12}. This procedure is thus followed for each $M_{1\,j}$ term until all of them have been determined.

Singular matrix is a *square matrix whose determinant is zero*. A determinant is zero if

1) any row or column in the matrix is zero;
2) any row or column, is equal to a linear combination of any other rows or columns.

Example:

$$A = \begin{bmatrix} 1 & 6 & 4 \\ 2 & 1 & 0 \\ 5 & -3 & -4 \end{bmatrix},$$

where row 1 = 3(row 2) − row 3. So, computing the determinant gives

$$\begin{aligned} |A| &= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}) \\ &= 1[1(-4) - 0(-3)] - 6[2(-4) - 0(5)] + 4[2(-3) - 1(5)] \\ &= -4 + 48 - 44 = 0. \end{aligned}$$

Degree of clustering symmetrically about the principal diagonal is another (of many) properties of a determinant. The more clustering there is, the higher the value of the determinant.

A1.5 Matrix "Division": Inverse Matrix

Matrix division must be thought of as multiplying by the inverse, as is true for scalar division,

$$\frac{x}{b} = xb^{-1} \tag{A1.27a}$$

and

$$bb^{-1} = 1. \tag{A1.27b}$$

Matrices are divided by applying the above – i.e., by multiplying by the inverse matrix. *Nonsingular square matrices* may have a unique inverse symbolized as

$$\mathbf{A}^{-1}$$

and

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}. \tag{A1.28}$$

The most basic inverse involves the use of cofactors and the determinant. First, recall that the definition of a cofactor, C_{ij}, is

$$C_{ij} = (-1)^{i+j}M_{ij}, \tag{A1.29}$$

where M_{ij} is the determinant of the submatrix created by eliminating the *i*th row and *j*th column of the matrix. Given this, the determinant of a matrix **A** is given by

$$\det A = a_{i1}C_{i1} + a_{i2}C_{i2} + \ldots + a_{in}C_{in}. \tag{A1.30}$$

Then, the *inverse is completely determined by knowledge of the cofactors*. In particular, $\mathbf{A}^{-1}$ is given as

$$\mathbf{A}^{-1} = \frac{\mathbf{C}^{T}}{\det A}, \tag{A1.31}$$

where $\mathbf{C}^{T}$ is the transpose of the **adjoint matrix** (or **adjugate** or **cofactor matrix**), which is defined as

$$C = \begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1n} \\ c_{21} & c_{22} & \ldots & c_{2n} \\ \vdots & & & \\ c_{m1} & c_{m2} & \ldots & c_{mn} \end{bmatrix}, \tag{A1.32}$$

and the determinant of the matrix **A** is also determined by combining the cofactors from the first row of the cofactor matrix, as seen in (A1.30).

Calculation of the inverse of a larger matrix is usually done using **elimination methods** (on the computer). These techniques (or methods used in place of

Box A1.4 Example of Inverse Using Adjoint

For a simple 2×2 matrix, the inverse is given by

$$A^{-1} = \frac{1}{\det\ A}\begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

according to (A1.31). So for the **A** matrix

$$A = \begin{bmatrix} 7 & 2 \\ 10 & 3 \end{bmatrix},$$

then

$$A^{-1} = \frac{1}{21-20}\begin{bmatrix} 3 & -2 \\ -10 & 7 \end{bmatrix} = \begin{bmatrix} 3 & -2 \\ -10 & 7 \end{bmatrix}$$

and

$$AA^{-1} = \begin{bmatrix} 7 & 2 \\ 10 & 3 \end{bmatrix}\begin{bmatrix} 3 & -2 \\ -10 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I.$$

inversion) will be presented when necessary, in context with specific data analysis techniques.

A1.5.1 Solution of Simultaneous Equations

A system of *n* simultaneous equations in *n* unknowns is easily handled by matrix algebra notation. For example, consider four equations in four unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4. \end{aligned} \tag{A1.31a}$$

This is written in matrix form as

$$\mathbf{Ax} = \mathbf{b}, \tag{A1.31b}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \text{coefficient matrix}$$

$$\mathrm{x} = \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \mathrm{x}_3 \\ \mathrm{x}_4 \end{bmatrix}; \ \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \\ \mathrm{b}_4 \end{bmatrix}.$$

This is easily solved (symbolically) using the simple operations of matrix algebra as

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{A1.32a}$$

(i.e., pre-multiply both sides by $\mathbf{A}^{-1}$; recall that order of multiplication is important). So,

$$\mathbf{I}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{A1.32b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{A1.32c}$$

That is, the unknown elements of **x** are simply given by pre-multiplying the **b** column vector with the inverse of the coefficient matrix, $\mathbf{A}^{-1}$.

Computational Considerations

The case of matrix inverses suggests that, while the above approach is extremely elegant symbolically, sometimes the computational effort may be considerable when dealing with large systems and a direct solution by elimination methods is quicker. This is revealed by consideration of the number of multiplications involved in a matrix inversion approach.

Solution by the inversion (as shown above) requires n^3 multiplications for the inversion and $n^2\ m$ more multiplications to finish the solution, where: n = number of equations per set and m = number of sets of equations (each of the same form, but different **b** matrix). Therefore, the total number of multiplications is: $n^3 + n^2\ m$.

Compare this to a direct solution by a typical elimination method (the standard way to solve a system of equations using algebra), which requires only $n^3/3 + n^2\ m$.

So, while the matrix form is easy to handle, one should not necessarily always use it blindly (we will consider many situations for which matrix solutions are ideal). Note that for sparse matrices, the above relationships may not hold; nor for symmetrical matrices or other special matrix forms in which case-special fast algorithms exist for the inversion. Also note that, for stability reasons, sometimes other special approaches are better suited for the solution of a large system. These will be introduced at the appropriate times.

A1.5.2 Additional Terms

Rank of a matrix is the number of linearly independent vectors the matrix contains (either row or column vectors).

Box A1.5 Example Solving Simple Ax = b

Consider solving the following two simultaneous equations (i.e., two equations in two unknowns):

$$\begin{aligned} 5x_1 + 7x_2 &= 19 \\ 3x_1 - 2x_2 &= -1. \end{aligned}$$

These represent the equations of lines, and the common solution represents the intersection of the two lines. In matrix form, these are given as

$$\begin{bmatrix} 5 & 7 \\ 3 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 19 \\ -1 \end{bmatrix}$$
$$\mathrm{A} \quad \mathrm{x} \quad = \quad \mathrm{b}.$$

To solve this requires the inverse of the **A** matrix:

$$\mathrm{A}^{-1} = \frac{1}{-10-21}\begin{bmatrix} -2 & -7 \\ -3 & 5 \end{bmatrix} = \begin{bmatrix} \frac{2}{31} & \frac{7}{31} \\ \frac{3}{31} & \frac{-5}{31} \end{bmatrix}.$$

Then,

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

and

$$\mathrm{A}^{-1}\mathrm{b} = \begin{bmatrix} \frac{2}{31} & \frac{7}{31} \\ \frac{3}{31} & \frac{-5}{31} \end{bmatrix} \begin{bmatrix} 19 \\ -1 \end{bmatrix} = \begin{bmatrix} \left(\frac{38}{31} - \frac{7}{31}\right) \\ \left(\frac{57}{31} + \frac{5}{31}\right) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

so the unknowns, $x_1 = 1$ and $x_2 = 2$ represent the solution and solve the system.

Rank has special meaning when dealing with systems of equations. Consider the common intersection of four lines of the standard form y = mx + b:

$$\begin{aligned} 2 &= \mathrm{m}1 + \mathrm{b} \\ 3 &= \mathrm{m}2 + \mathrm{b} \\ 4 &= \mathrm{m}3 + \mathrm{b} \\ 5 &= \mathrm{m}4 + \mathrm{b}, \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{m} \\ \mathrm{b} \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}.$$

Box A1.5 Example of Matrix Rank

Consider

$$A = \begin{bmatrix} 1 & 4 & 0 & 2 \\ 1 & 0 & 1 & -1 \\ -3 & -4 & -2 & 0 \end{bmatrix}.$$

Since

$$\text{row } 3 = -(\text{row } 1) - 2(\text{row}2)$$

or

$$\text{col } 3 = \text{col } 1 - \frac{1}{4}(\text{col } 2)$$

and

$$\text{col } 4 = -(\text{col } 1) + \frac{3}{4}(\text{col } 2),$$

the matrix **A** has rank 2 (i.e., it has only two linearly independent vectors, independent of whether it is viewed by rows or columns).

Since the a matrix is order 4×2, the largest rank it can possibly have is 2 (obviously, there cannot be more linearly independent column or row vectors than the smallest number of columns or rows). This says that at least two of the rows (= equations) are dependent, and thus provide duplicate information. Inspection of the above matrix shows that row (4) = (row 2) + (row 3) − (row 1).

A matrix is **rank deficient** if its rank < min(*m,n*) and it is of **full rank** if its rank = min (*m,n*).

The **rank of a matrix product** must be less than or equal to the smallest rank of the matrices being multiplied, so

$$A_{(\text{rank } 2)}B_{(\text{rank } 1)} = C_{(\text{rank } 1)}.$$

This seemingly esoteric fact is actually extremely helpful when considered from another angle. That is, *if a matrix has rank r, then any matrix factor of it must have rank of at least r*. Since the rank cannot be greater than the smallest of *m* or *n*, in a *m*× *n* matrix, this definition also limits the size (order) of factor matrices. So you cannot factor a matrix of rank 2 into two matrices of which either is of less than rank 2, so *m* and *n* of each factor must also be 2.

Rank of a matrix **A** is most easily determined by performing an SVD decomposition on the matrix and counting the number of nonzero singular values (= rank) in the diagonal matrix of the SVD decomposition. If there are no zeros, matrix **A** is of **full rank**.

Minor is the determinant of an equal number of rows and columns from a (encompassing) matrix. If a matrix has rank r, then there must be at least one nonzero minor of order r present in the matrix (and no nonzero minors of order > r). Remember that a determinant of a matrix is zero if its rank is not equal to its order (i.e., if any row or column in the matrix is linearly dependent upon any other rows or columns). Unfortunately, we cannot compute the rank of a matrix by examining for nonzero determinants do to machine round-off errors.

The **trace** of a square matrix is simply the sum of the elements along the principal diagonal. It is symbolized as tr**A1.** This property is useful in calculating various quantities from matrices.

Submatrices are smaller matrix partitions of a larger, encompassing supermatrix. For example,

$$\begin{bmatrix} \text{supermatrix} \\ \mathbf{F} \end{bmatrix} = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \quad \text{Submatrices :}$$

Such partitioning is frequently be useful.

Major product moment is the *square matrix* resulting from the product $\mathbf{AA}^T$ in which the diagonal elements are the sum of squares for each row of the matrix.

$$\mathbf{A}\begin{bmatrix} 1 & -3 \\ 2 & 0 \\ 2 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 2 \\ -3 & 0 & 2 \end{bmatrix}\mathbf{A}^T$$

$$\begin{bmatrix} 10 & 2 & -4 \\ 2 & 4 & 4 \\ -4 & 4 & 8 \end{bmatrix}\mathbf{AA}^T$$

Minor product moment is the *square matrix* resulting from the product $\mathbf{A}^T\mathbf{A}$ in which the diagonal elements are the sum of squares for each column of the matrix.

A1.6 Useful Properties

$$(A^T)^T = A \tag{A1.33}$$

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A} \tag{A1.34}$$

$$(A^{-1})^T = (A^T)^{-1} = A^{-T} \tag{A1.35}$$

So, if

$$\mathbf{D} = \mathbf{ABC}, \tag{A1.36a}$$

then

$$\mathbf{D}^{-1} = \mathbf{C}^{-1}\mathbf{B}^{-1}\mathbf{A}^{-1} \tag{A1.36b}$$

$$\mathbf{D}^{\mathrm{T}} = \mathbf{C}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}. \tag{A1.36c}$$

These last two "*reversal rules*" for inverse and transpose products are constantly needed for manipulation of matrix equations and are also useful for eliminating or minimizing the number of matrix inverses or transposes requiring calculation.

A1.7 More Advanced Topics

While the preceding gives an introduction to the manipulative capabilities of matrix algebra, the topic encompasses considerably more. In particular, linear algebra represents a different means through which most of higher mathematics can be viewed. Linear algebra provides a framework for thinking about the typical calculations we will encounter that tends to unify the otherwise diverse topics to be discussed. This section presents an overview of the elements of linear algebra that define this framework.

A1.7.1 Vector Space and Basis

Consider a set of *m-element* vectors $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ belonging to **real vector space** $\mathbf{R}^{\boldsymbol{m}}$. A collection of n vectors $\mathbf{x_j} \in \mathbf{R}^{\boldsymbol{m}}$, along with every linear combination of those n vectors, forms a subspace defined as the **span** of $\mathbf{x_j}$**.** That is, they *span* the space because they can produce every *m-dimensional* vector in that space via some linear combination. If the set of vectors span the space and are independent, that is $\sum_{i=1}^{n} a_i \mathbf{x}_i \neq 0$ for all $a_1 \neq 0$, the set represents a *basis*. With this basis, every *m-element* vector in the space can be generated and the number of vectors in the basis is the dimension of the *space* (this **implies that** $m = n$). From our perspective, if we wish to describe (e.g., interpolate, generate, or otherwise form) any vector of m elements, we must do so using an *m-dimensional* basis, ensuring that we can fit *any* such vector, or, alternatively, allowing us to solve $\mathbf{Ax} = \mathbf{b}$ for any vector **b.** So, with this set of basic vectors, we can describe any vector of the same dimension.

The **null space** of a matrix **A**, denoted N(**A**), is defined as all vectors **x**, such that $\mathbf{Ax} = \mathbf{0}$. In systems of equations, this represents the homogeneous system of equations (no source or sink terms). In linear algebra, it represents the following operation between the elements of **x** and the column vectors contained within **A**:

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ \vdots \\ a_{n2} \end{bmatrix} + \leftrightharpoons + x_m \begin{bmatrix} a_{1m} \\ a_{2m} \\ a_{3m} \\ \vdots \\ a_{nm} \end{bmatrix} = x_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{A1.37}$$

This implies that the linear combination of column vectors (in **A**), with the appropriate constants (the elements of **x**), can be summed linearly to equal zero. For this to happen, the columns of **A** must be linearly dependent, unless the only vector **x** in the nullspace is the null vector – that is, the only way to add the columns of **A** together to get the zero (null) vector is when you multiply each column by zero (the trivial case).[1]

The **row space** of a matrix is the collection of all possible linear combinations of the row vectors in the matrix. Similarly, the **column space** of a matrix is the collection of all possible linear combinations of the column vectors in the matrix.

A1.7.2 Vector Differentiation

Direct differential manipulation of the vectors and matrix equations facilitates many computations. This is easily done, given the following convention for vector differentiation:

$$\frac{\partial}{\partial x}(\mathbf{Ax}) \equiv \frac{\partial}{\partial x}(\mathbf{Ax})^{\mathrm{T}} = \frac{\partial}{\partial x}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}. \tag{A1.38}$$

For expressions often encountered in statistical applications, such as

$$\frac{\partial}{\partial x}(\mathrm{x}^{\mathrm{T}}\mathbf{Ax}) \equiv \frac{\partial}{\partial x}(\mathrm{x}^{\mathrm{T}}\mathbf{Ax})^{\mathrm{T}} = \frac{\partial}{\partial x}(\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}x)., \tag{A1.39}$$

use the principle of differentiation of parts to give[2]

$$\begin{aligned}\frac{\partial}{\partial x}(\mathrm{x}^{\mathrm{T}}\mathbf{Ax}) &= \frac{\partial}{\partial x}\mathrm{x}^{\mathrm{T}}(\mathbf{Ax})^{\mathrm{T}} = \frac{\partial}{\partial x}\mathbf{x}^{\mathrm{T}}(\mathbf{A}^{\mathrm{T}}\mathrm{x}) \\ &= \mathbf{Ax} + \mathbf{A}^{\mathrm{T}}\mathbf{x}.\end{aligned} \tag{A1.40}$$

A1.7.3 Quadratic Form

Up until now we have dealt with strictly linear forms of systems. Now we wish to deal with quadratic forms – in particular, $\mathbf{x}^{\mathrm{T}}\mathbf{Ax}$.

First we must establish some necessary terminology. Consider a vector product, $\mathbf{x}^{\mathrm{T}}\mathbf{Ax}$, where **A** is square. This product represents a quadratic form of the vector **x**. Basically, it

[1] Recall the definition of linear dependency: a1x1 + a2x2 + … + anxn = 0.

[2] This rule can be somewhat confusing. To help, set $\mathbf{x}^{\mathrm{T}} = \mathbf{y}$, then rewrite $\mathbf{x}^{\mathrm{T}}\mathbf{Ax} = \mathbf{yAx}$. Now, consider differentiation by parts for scalars: $\frac{\partial}{\partial x}(\mathbf{yAx}) = \mathbf{Ax}\frac{\partial}{\partial x}\mathbf{y} + \mathbf{yA}\frac{\partial}{\partial x}\mathbf{x}$. However, for vectors, the order of operations is important. Therefore, we take account of the equivalency given in (A.38), which states that the derivatives of $\mathbf{x}^{\mathrm{T}}\mathbf{Ax}$ and $\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{x}$ are equal. We use the first form to treat **Ax** as being independent of **x**, so we form the counterpart to $\mathbf{Ax}\frac{\partial}{\partial x}\mathrm{y}$ as $(\frac{\partial}{\partial x}\mathrm{x}^{\mathrm{T}})\mathrm{A} = \mathrm{A}$. Having done this, we must now treat $\mathbf{x}^{\mathrm{T}}\mathbf{A}$ as being independent of x, which we do by using the second form, in which case the $\mathbf{x}^{\mathrm{T}}\mathbf{A}$ has been transposed and reversed to $\mathbf{A}^{\mathrm{T}}\mathbf{x}$. With this, the derivative can again be written in the proper form for vector differentiation, where the counterpart to $\mathbf{yA}\frac{\partial}{\partial x}\mathbf{x}$ *is* $(\frac{\partial}{\partial x}\mathbf{x}^{\mathrm{T}})\mathbf{B} = \mathbf{B}$, where $\mathbf{B} = \mathbf{A}^{\mathrm{T}}\mathbf{x}$. In this manner, we have accomplished the product expansion while satisfying the convention for the proper form of vector differentiation.

represents the weighted sum of squares and all cross-products (i.e., the $x_i x_j$, where $i \neq j$), where the weights are the various elements of matrix **A**. Explicitly,

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j \tag{A1.38a}$$

$$= \sum_{i=1}^{n} a_{ii}x_i^2 + \sum_{i=1}^{n}\sum_{j\neq 1}^{j=1^{n}} a_{ij}x_i x_j \tag{A1.38b}$$

$$= \sum_{i=1}^{n} a_{ii}x_i^2 + \sum_{i=1}^{n}\sum_{j\geq 1}^{j=1^{n}} (a_{ij}+a_{ij})x_i x_j. \tag{A1.38c}$$

Inspection of (A1.38c), reveals that the exact same quadratic sum could be represented using a different square matrix, **A**. For example, $\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{B}\mathbf{x}$, as long as the diagonal elements and the sum of $(a_{ij} + a_{ij})$ (that is, the sum of symmetrical, off-diagonal pairs of both **A** and **B** are equal).[3] Since this can be satisfied by an infinite number of matrices, there is no single unique matrix **A** for this quadratic form of **x**.

For example,

$$A = \begin{bmatrix} 1 & 4 & 2 \\ 2 & 3 & 3 \\ 2 & 5 & 1 \end{bmatrix};\ B = \begin{bmatrix} 1 & 1 & 1 \\ 5 & 3 & 1 \\ 3 & 7 & 1 \end{bmatrix};\ \text{and } C = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 3 & 4 \\ 2 & 4 & 1 \end{bmatrix} \tag{A1.39}$$

all yield the same quadratic sum, so the quadratic forms $\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{B}\mathbf{x} = \mathbf{x}^T\mathbf{C}\mathbf{x}$. More importantly, notice that **C** is a symmetrical matrix. Therefore, we are sacrificing nothing by stipulating that the coefficient matrix, **A**, in the quadratic form is symmetric. Since there is only *one* symmetrical matrix that satisfies this equality, it is unique. The symmetrical version of any matrix **A** satisfying the quadratic form can always be obtained as (**A** + **A**T)/2.

Using the symmetric form, (A1.38c) can be rewritten as

$$X^T Ax = \sum_{i=1}^{n} a_{ii}x_i^2 + 2\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j>1}}^{n} a_{ij}x_i x_j \tag{A1.40}$$

Because of the uniqueness of the symmetrical form, without loss of generality, and because of the special properties of symmetric matrices, we will always restrict the quadratic form to involve the unique symmetric coefficient matrix, **A** (so **A** = **A**T).

Positive definite. In the simplest quadratic form case, where **A** = **I**, the quadratic sum reduces to $\mathbf{x}^T\mathbf{x}$, the sum of squares of the x values. This scalar product is a positive, nonzero number for all **x** except **x** = 0. When the matrix **A** is not the identity matrix, its elements now represent weights against which the squares and cross-product terms will

[3] That is, these pairs consist of an element plus the element that occupies the same position of the current element in the transpose of the matrix. That is, the aij and aji.

be scaled before summing. In this case, the quadratic sum need not sum to positive values as it must when $\mathbf{A} = \mathbf{I}$. However, in those cases where

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \text{ (all } \mathbf{x} \neq \mathbf{0}\text{)}, \tag{A1.41}$$

that is, where the quadratic sum is greater than zero for all $\mathbf{x} > 0$), the quadratic form is called **positive definite**, and the corresponding (symmetric) coefficient matrix, **A**, is a **positive definite matrix**.

If

$$\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0 \text{ (all } \mathbf{x} \neq \mathbf{0}; \mathbf{x}^T\mathbf{A}\mathbf{x} = 0 \text{ for some } \mathbf{x} \neq \mathbf{0}\text{)}, \tag{A1.42}$$

then the quadratic form is called **positive semidefinite** and the corresponding (symmetric) coefficient matrix, **A**, is a **positive semidefinite matrix.**[4]

The quadratic form is most commonly encountered when dealing with variance and covariance matrices. For example, consider writing the sample variance[5] in its quadratic form. First, expand the sum of squared deviations about the mean:

$$\begin{aligned}\sum_{i=1}^{n}(x_i-\bar{x})^2 &= \sum_{i=1}^{n}x_i^2 - n\bar{x}^2 \\ &= \sum_{i=1}^{n}x_i^2 - n\left[\frac{1}{n}\sum_{j=1}^{n}x_j\right]^2 \\ &= \sum_{i=1}^{n}x_i^2 - \sum_{j=1}^{n}\frac{1}{n}x_i^2 - \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\frac{1}{n}x_ix_j \\ &= \sum_{i=1}^{n}\left(\frac{n-1}{n}x_i^2\right) - \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\frac{1}{n}x_ix_j \end{aligned} \tag{A1.43a}$$

Now multiply through by $1/(n-1)$ to complete the operation:

$$= \left[\sum_{i=1}^{n}\left(\frac{1}{n}x_i^2\right) - \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\left(\frac{1}{n(n-1)}x_ix_j\right)\right] \tag{A1.43b}$$

$$= \mathbf{x}^T\mathbf{C}\mathbf{x}. \tag{A1.43c}$$

The expanded sum (A1.43b) is in the form of (A1.38b) with the $a_{ii} = 1/n$ and the $a_{ij} = -1/([n(n-1)]$. Therefore, in the quadratic equivalent, $\mathbf{x}^T = (x_1\ x_2 \ldots x_n)$ and $\mathbf{C} = [(n+2)/n(n-1)]\mathbf{I} - [1/[n(n-1)]\mathbf{J}_n$, $\mathbf{J}_n$ is a square unity matrix (all elements equal to one). Because the

[4] Note that some people like to abbreviate a positive definite and positive semidefinite matrix as **pd** and **psd**, respectively. I will avoid this notation, since I use the abbreviation psd to represent power spectral density. Regardless, the usage for the two representations of psd are so different that there should never be any confusion as to which meaning is intended. Also, some people use **non-negative definite** (**nnd**) for both positive definite and positive semidefinite,while others use non-negative definite to mean positive semidefinite only.

[5] The biased variance results when dividing the sum of squared perturbations about the mean by n, instead of n − 1 (which produces the unbiased estimate). Note that for the present example, the result would differ only in that the factor $1/n^2$ would become $1/[n(n-1)]$.

matrix **C** serves to remove the mean (while also scaling), it is often referred to as the **centering matrix**.

Pseudo-inverse. Pseudo-inverses are the means for computing the "inverse" of a nonsquare matrix, **A**. Specifically, if **A** is overdetermined, having more rows than columns, then a pseudo-inverse known as the **left inverse** is obtained as

$$\mathbf{A}_{\mathrm{L}}^{-1} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}. \tag{A1.44a}$$

and for the case where **A** is underdetermined, having more columns than rows, then a pseudo-inverse known as the **right inverse** is obtained as

$$\mathbf{A}_{\mathrm{R}}^{-1} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}. \tag{A1.44b}$$

For equations of the form: $\mathbf{Ax} = \mathbf{b}$, these lead to the following solutions (discussed more in context later).

For overdetermined systems, where matrix **A** has more rows than columns, the inverse solution (in fact, the least-squares solution) is thus

$$\mathbf{x} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b} = \mathbf{A}_{\mathrm{L}}{}^{-1}\mathbf{b}. \tag{A1.45a}$$

whereas for underdetermined systems, where matrix **A** has more columns than rows, the inverse solution is

$$\mathbf{x} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{b}. \tag{A1.45b}$$

The implications of these are discussed more fully in the regression discussions.Using SVD (discussed below), the pseudo-inverse is:

$$\mathbf{VS^*UT} \tag{A1.45c}$$

Where **S* is the original diagonal matrix S**, with each element inverted (i.e., σ^{11}becomes $1/\sigma^{11}$).

A1.7.4 Ill-Conditioning

Ill-conditioned systems are those in which a *small change* in one or more of the coefficient values (in the **A** matrix) or observed values (in the **b** vector) lead to a large change in the solution of the system unknowns. Ill-conditioning arises from three distinctly different sources:

- The physical problem itself may represent an unstable system, such as a bicycle balancing on a tightrope or a pencil balancing on its eraser.
- The basis chosen to represent the system may be poorly chosen, such as a polynomial basis over a range of 0 to 1, versus a basis that is orthogonal over this range.
- The method of solution may be sensitive to small changes, as are most matrix inversion techniques and pivotal methods based on elimination procedures under certain conditions.

Here we consider only the latter two sources, since they are the more common in curve fitting (interpolation and smoothing) problems. Conceptually, they can both be described in a similar manner.

Consider the solutions to the interpolation and least-squares problems that require inversion of $\mathbf{A}$ or the $\mathbf{A}^T\mathbf{A}$ matrix products. A square matrix, $\mathbf{A}$, can only have an inverse if it is of full rank (i.e., there are as many *independent* vectors in the matrix as there are columns or rows). If this is not the case, the determinant of the matrix is zero, and $\mathbf{A}$ cannot be inverted – recall that the elements of $\mathbf{A}^{-1}$ are scaled by 1/(det $\mathbf{A}$), which, for det $\mathbf{A} = 0$, is singular (a square matrix that has its determinant equal to 0 is known as a *singular matrix*). If the vectors in $\mathbf{A}$ are *almost* dependent, then the value of det $\mathbf{A}$ is small in magnitude relative to certain of the matrix cofactors. This can result in a round-off problem, leading to the loss of accuracy.

The following two examples (from Jennings, 1977), demonstrate the problem. First, consider the system $\mathbf{Ax} = \mathbf{b}$ for which the coefficient matrix has the following elements:

$$\mathbf{A} = \begin{bmatrix} .5 & (.5+\alpha) \\ .5 & (.5+\alpha) \\ .5 & (.5-\alpha) \\ .5 & (.5-\alpha) \end{bmatrix}. \tag{A1.46}$$

This matrix has rank 2 and the square matrix $\mathbf{A}^T\mathbf{A}$ is

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1+4\alpha^2 \end{bmatrix}. \tag{A1.47}$$

This matrix product is square and of full rank, and thus has an inverse. However, if $\alpha \ll 1$, then $\alpha^2 \ll \alpha$. Machine round-off may force the computer to represent the small $4\alpha^2$ as 0. This will artificially make the matrix product in (A1.47) singular, with no matrix solution. Conceptually, the source of the problem is that the two rows (or columns) in $\mathbf{A}^T\mathbf{A}$ are nearly linearly dependent when α is small.

Note that the magnitude of the determinant is sometimes (erroneously) used as a measure of the **condition** of the matrix – the smaller the magnitude, the worse the conditioning, because the value of 1/(det $\mathbf{A}$) will approach singularity. This can be shown to be invalid through a simple scaling experiment similar to the above example. That is, one can scale the above $\mathbf{A}^T\mathbf{A}$ matrix so that its determinant is changed tremendously, yet the condition of the matrix is not changed at all. There are actually more accurate measures of the condition of a matrix that reveal whether it is ill-conditioned or not, but they can be computationally intensive so people often avoid them.

Second, consider the exact fit to n data points using an *n*th-degree polynomial or a least-squares fit to these data of an (m − 1)th-degree polynomial, where m is a large number. So,

$$y_i = a_1 + a_2 x_i + a_3 x_1^{\,2} + a_4 x_1^{\,3} + \ldots, \tag{A1.48}$$

where the x_i are evenly spaced and span the range from 0 to 1.

If n is large, the coefficient matrix, $\mathbf{A}$, can be very ill-conditioned (especially if scaled by 1/n, in which case the matrix approximates the ill-conditioned *Hilbert matrix*). This is

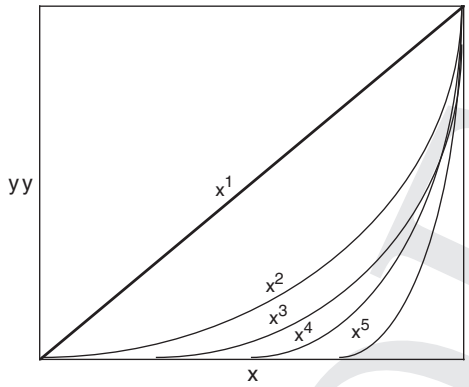


Figure A1.5 Polynomials of different degrees, showing how the higher degrees start to look similar at larger y values.

because the terms of the polynomial basis, x, x^2, … x^n have similar shape and thus display a strong degree of linear dependence, as seen in Figure A1.5.

This problem arises due to the choice of the basis for the given range of x. In this case, a simple solution exists if you choose another basis that has constituent functions that are strongly independent over the range of x – or better, that are orthogonal over the range (e.g., Chebyshev polynomials or orthogonal polynomials).

Alternatively, another solution is to convert the offending functions into a mutually orthogonal set through a set of simple operations on the system. This approach works equally well for both of the above two examples.

A1.7.5 Orthogonal Decomposition

Orthogonal decomposition is the name given to general procedures used to transform the variables into a set of mutually orthogonal functions. Since the orthogonal functions are "completely" independent, the above problems of ill-conditioning are largely circumvented.

Orthogonal decomposition relies on the identity that any $n \times m$ matrix **A** ($n \geq m$) can be decomposed into the product of an $n \times m$ orthogonal matrix **Q** satisfying

$$Q^T Q = I, \tag{A1.49}$$

and another matrix or matrix product, such as an $m \times m$ upper-triangular matrix **R**, so

$$A_{nm} = Q_{nm} R_{mm}. \tag{A1.50}$$

This is called **QR decomposition** and is one form of orthogonal decomposition.

Singular-value decomposition (**SVD**) is another popular form where

$$A_{nm} = U_{nn} S_{nm} V^T{}_{mm} \tag{A1.51}$$

and **U, V**^{T} are orthogonal and **S** is a diagonal matrix.

The matrices **Q, U** and **V** are all **orthonormal**. This means that each row (or column) has magnitude 1, so ||q|| = 1.

Both QR decomposition and SVD have particular properties that make them advantageous for a variety of matrix problems. Regarding curve fitting problems, the QR decomposition is often considered ideal for problems in which n ˜100, though Lawson and Hanson (1974) like SVD in most cases because it returns information regarding the coefficient and data matrices.

I will demonstrate QR decomposition here because it is the more straightforward.

Given the QR decomposition, a system,

$$\mathbf{Ax} = \mathbf{b}, \tag{A1.52}$$

is decomposed to

$$\mathbf{QRx} = \mathbf{b} \tag{A1.53a}$$

$$\mathrm{Q^TQRx} = \mathrm{Q^Tb}, \tag{A1.53b}$$

and, since $\mathbf{Q^TQ} = \mathbf{I}$,

$$\mathrm{Rx} = \mathrm{Q^Tb}. \tag{A1.54}$$

R is nonsingular, so this equation can always be solved (the implication of the solution when **A** is not square as discussed below). Furthermore, this matrix equation is easily solved using **backsubstitution** – a simple procedure solving each consecutive row of **Rx**.

The **Rx** system represents a system of the form

$$\begin{aligned} 2x_1 + 4x_2 - 3x_3 &= y_1 \\ 1x_2 + 2x_3 &= y_2 \\ x_3 &= y_3, \end{aligned} \tag{A1.55}$$

which is solved from the bottom up (by backsubstituting):

$$\begin{aligned} x_3 &= y_3 \\ x_2 &= (y_2 - 2x_3)/1 \\ x_1 &= (y_1 + 3x_3 - 4x_2)/2. \end{aligned} \tag{A1.56}$$

Now consider the QR decomposition given a nonsquare matrix **A**, so

$$\mathbf{Ax} = \mathbf{b} \tag{A1.57}$$

and the standard least-squares solution is given by

$$[\mathbf{A^TA}]^{-1}\mathbf{A^TAx} = [\mathbf{A^TA}]^{-1}\mathbf{A^Tb}. \tag{A1.58}$$

Multiplying the least-squares solution, (A1.58), by $[\mathbf{A^TA}]$ to get

$$\mathbf{A^TAx} = \mathbf{A^Tb} \tag{A1.59}$$

and substituting $\mathbf{A} = \mathbf{QR}$ and $\mathbf{A^T} = \mathbf{R^TQ^T}$ (employing the reversal rule of transposed products) gives

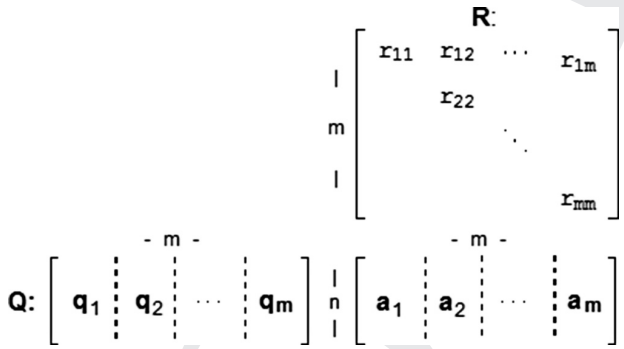
$$\mathbf{R^TQ^TQRx} = \mathbf{R^TQ^Tb}, \tag{A1.60}$$

Box DA1.1 Derivation of Orthogonal Decomposition (Optional)

DA1.1.1 Gram–Schmidt Procedure

Orthogonal transformations (e.g., computing the **Q** and **R** matrices) are most easily demonstrated using a method known as the **Gram–Schmidt procedure**, though the **Householder transform** is more efficient (and popular), as is the **Givens transformation**.[6]

The Gram–Schmidt Procedure works as follows:Consider the product **QR** = **A**,



where the **Q** and **A** matrices have been partitioned into m column vectors, so

$$a_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}; \quad q_1 \begin{bmatrix} q_{11} \\ q_{21} \\ q_{31} \\ \vdots \\ q_{n1} \end{bmatrix}. \tag{DA1.1.2}$$

So, the multiplication **QR** can be done in terms of vectors. The first vector of **A** is

$$\mathbf{a}_1 = \mathbf{q}_1 r_{11} = r_{11}\mathbf{q}_1. \tag{DA1.1.3}$$

since r_{11} is a scalar.

Now consider the **orthonormal property** of **Q**. This states that

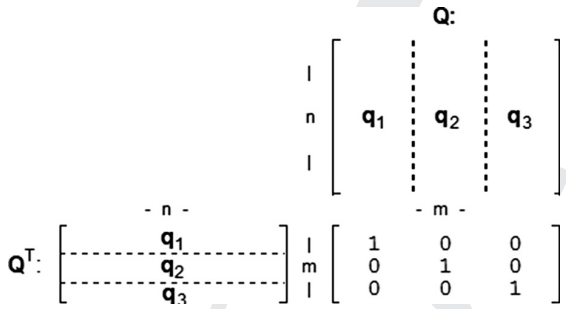
$$q_i^T q_j = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases} \tag{DA1.1.4}$$

In other words, (DA1.1.4) states that the magnitude (L_2 norm, or Euclidean length) of each vector within **Q** is unity. For example, $[q_{11}^2 + q_{12}^2 + \ldots + qq_{n1}^2]^{1/2} = \mathbf{q}_1^T\mathbf{q}_1 = 1$. Also, each vector within **Q** is orthogonal to every other vector in **Q**, hence, by definition, the dot product between all of the vectors is zero.

[6] A popular method for least-squares (L2 norm) type problems involves Golub's method, using Householder transformations.

Box DA1.1 (cont.)

Consider the case of three columns in **Q**, so m = 3:



so $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$.

Multiplying (DA1.1.3) by a_1^T and then imposing the orthonormal constraint, $\mathbf{q}^{\mathbf{T}}\mathbf{q}_1 = 1$ gives

$$\begin{aligned}\mathbf{a}_1^{\mathrm{T}}\mathbf{a}1 &= (\mathbf{q}_1 r_{11})^{\mathrm{T}}(\mathbf{q}_1 r_{11}) \\ &= r_{11}\mathbf{q}_1^{\mathrm{T}}\mathbf{q}_1 r_{11} \\ &= r_{11}^2.\end{aligned} \tag{DA1.1.6}$$

so

$$r_{11} = (\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1)^{1/2} = \|\mathbf{a}_1\|, \tag{DA1.1.7}$$

which is the Euclidean norm of the vector $\mathbf{a}_1$. Given r_{11}, (DA1.1.3) can be solved for $\mathbf{q}_1$:

$$\mathbf{q}_1 = \frac{1}{r_{11}}\mathbf{a}1. \tag{DA1.1.8}$$

So, $\mathbf{q}_1$ is simply the $\mathbf{a}_1$ vector normalized by its Euclidean norm (scaling a vector by its norm is known as **normalization** – it is a standard procedure used to make the norm of any vector unity).

The second vector of **A,** $\mathbf{a}_2$, is

$$\mathbf{a}_2 = \mathbf{q}_1 r_{12} + \mathbf{q}_2 r_{22}. \tag{DA1.1.9}$$

Pre-multiplying by $\mathbf{q}_1^{\mathrm{T}}$,

$$\mathbf{q}_1^{\mathrm{T}}\mathbf{a}_2 = \mathbf{q}_1^{\mathrm{T}}\mathbf{q}_1 r_{12} + \mathbf{q}_1{}^{\mathrm{T}}\mathbf{q}_2 r_{22}, \tag{DA1.1.10}$$

and imposing orthonormal conditions (i.e., $\mathbf{q}_1^{\mathrm{T}}\mathbf{q}_1 = 1$ and $\mathbf{q}_{1\mathrm{T}}\mathbf{q}_2 = 0$) solves (DA1.1.10) for r_{12}:

$$\mathbf{q}_1^{\mathrm{T}}\mathbf{a}_2 = r_{12}. \tag{DA1.1.11}$$

Rearranging equation (DA1.1.9) in terms of the two unknowns, $\mathbf{q}_2$ and r_{22}, gives

$$\mathbf{q}_2 r_{22} = \mathbf{a}_2 - \mathbf{q}_1 r_{12}. \tag{DA1.1.12}$$

This is abbreviated by defining the known quantity as

$$\widetilde{\mathbf{a}}_2 = \mathbf{a}_2 - \mathbf{q}_1 r_{12} \tag{DA1.1.13a}$$

Box DA1.1 (cont.)

and, substituting (DA1.1.12) into (DA1.1.13a), gives

$$\widetilde{\mathbf{a}}_2 = \mathbf{q}_2 \mathrm{r}_{22} \tag{DA1.1.13b}$$

With this, we can follow a solution procedure similar to that used to solve (DA1.1.3):

$$\begin{aligned} \widetilde{\mathbf{a}}_2^{\mathrm{T}} \widetilde{\mathbf{a}}_2 &= \mathrm{r}_{22} \mathbf{q}_2^{\mathrm{T}} \mathbf{q}_2 \mathrm{r}_{22} \\ &= r_{22}^2 \end{aligned} \tag{DA1.1.14}$$

so

$$\mathrm{r}_{22} = (\widetilde{\mathbf{a}}_2^{\mathrm{T}} \widetilde{\mathbf{a}}_2)^{1/2} = ||\widetilde{\mathbf{a}}_2|| \tag{DA1.1.15}$$

Finally, substitution of (DA1.1.15) into (DA1.1.13b) gives $\mathbf{q}_2$:

$$\begin{aligned} \mathbf{q}_{22} &= \frac{1}{\mathrm{r}_{22}} \widetilde{\mathbf{a}}_2 \\ &= \frac{1}{\mathrm{r}_{22}} (\mathbf{a}_{2-} \mathbf{q}_1 r_{12}). \end{aligned} \tag{DA1.1.16}$$

This simple procedure is repeated until all elements of **R** and columns of **Q** are formed, thus transforming **A** to **QR**. Other approaches can also be used to efficiently obtain the QR decomposition besides the Gram–Schmidt procedure, which is relatively inefficient. (Note that the **R** matrix is identical to the upper-triangular matrix obtained by Cholesky decomposition, though the latter can suffer from round-off accumulation error.)

DA1.1 Overview of Householder Transformation

The **Householder transformation** is known as a **reflection transformation**. It is an efficient and popular method for actually computing the QR decomposition, though it serves a variety of other uses as well. It is a **unitary transformation,** meaning that it *preserves length* (magnitude), so if the transformation matrix M is unitary, then

$$\mathbf{M}^{\mathrm{T}} \mathbf{M} = \mathbf{I}, \tag{DA1.1.17}$$

and **Mx** and **x** have the same length:

$$\begin{aligned} \mathbf{x}^{\mathrm{T}} \mathbf{x} &= (\mathbf{M}\mathbf{x})^{\mathrm{T}} \mathbf{M} x \\ &= \mathbf{x}^{\mathrm{T}} \mathbf{M}^{\mathrm{T}} \mathbf{M} \mathbf{x} \\ &= \mathbf{x}^{\mathrm{T}} \mathbf{x}. \end{aligned} \tag{DA1.1.18}$$

The **Householder transformation matrix, H**, transforms the coefficient matrix, **A**, to an upper diagonal form (as all QR decompositions do).

Box DA1.1 (cont.)

So,

$$\mathbf{A} = \mathbf{QR} \tag{DA1.1.19}$$

and, from condition (DA1.1.17),

$$\mathbf{Q}^{\mathrm{T}}\mathbf{A} = \mathbf{R}, \tag{DA1.1.20}$$

or, in terms of the Householder transform matrix, **H**,

$$\mathbf{HA} = \mathbf{R}. \tag{DA1.1.21}$$

Thus **H** (= $\mathbf{Q}^{\mathrm{T}}$) *transforms* **A** to **R** and the length of **A** is preserved.

The Householder transformation accomplishes the transform to **R** by transforming each column vector, $\mathbf{a}_j$, in **A**, one at a time,

$$\mathbf{Ha}_j = -\sigma||\mathbf{a}_j||\mathbf{e}_1, \tag{DA1.1.22}$$

where

$$\mathrm{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \; \sigma = \begin{cases} +1 & \mathrm{a}_{1j} \geq 0 \\ -1 & \mathrm{a}_{1j} < 0 \end{cases}, \tag{DA1.1.23}$$

so the sign of the transformed column simply reflects the sign of the first element in the column vector being transformed.

The transformation of $\mathbf{a}_j$ is accomplished by defining an arbitrary column vector:

$$\mathbf{v} = \mathbf{a}_j + \sigma||\mathbf{a}_j||\mathbf{e}_1. \tag{DA1.1.24}$$

Then,

$$\mathrm{H} = \mathrm{I}_m - \frac{2\mathrm{vv}^{\mathrm{T}}}{\mathrm{v}^{\mathrm{T}}\mathrm{v}}. \tag{DA1.1.25}$$

Note that while $\mathbf{v}^{\mathrm{T}}\mathbf{v}$ = scalar (the dot product); $\mathbf{vv}^{\mathrm{T}}$ is an $m \times m$ matrix (where m is the size of the **v** column vector):

$$\mathrm{vv}^{\mathrm{T}} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_m \\ \mathrm{a}_1^2 & \mathrm{a}_1\mathrm{a}_2 & \cdots & \mathrm{a}_1\mathrm{a}_m \\ \mathrm{a}_2\mathrm{a}_1 & \mathrm{a}_2^2 & & \\ \vdots & & \ddots & \\ \mathrm{a}_m\mathrm{a}_1 & \mathrm{a}_m\mathrm{a}_1 & & \mathrm{a}_m^2 \end{bmatrix}. \tag{DA1.1.26}$$

A **reflection matrix** such as **H,** above, is its own inverse, so

$$\mathbf{H} = \mathbf{H}^{-1} \tag{DA1.1.27}$$

Box DA1.1 (cont.)

and

$$\mathbf{H}^{-1}\mathbf{H} = \mathbf{I} = \mathbf{H}^{\mathrm{T}}\mathbf{H}, \tag{DA1.1.28}$$

since

$$\begin{aligned} \mathrm{H}^{\mathrm{T}}\mathrm{H} &= \left(\mathrm{I} - \frac{2\mathrm{v}\mathrm{v}^{\mathrm{T}}}{\mathrm{v}^{\mathrm{T}}\mathrm{v}}\right)^2 \\ &= \mathrm{I} - \frac{4\mathrm{v}\mathrm{v}^{\mathrm{T}}}{\mathrm{v}^{\mathrm{T}}\mathrm{v}} + \frac{4\mathrm{v}(\mathrm{v}\mathrm{v}^{\mathrm{T}})\mathrm{v}^{\mathrm{T}}}{(\mathrm{v}^{\mathrm{T}}\mathrm{v})^2} \\ &= \mathrm{I}. \end{aligned} \tag{DA1.1.29}$$

Pre-multiplication by **H** converts to zeros all of the elements *below* the main diagonal of one column of the matrix being transformed. *Post-multiplication* zeros the elements *above* the main diagonal of one row of the matrix. Thus, depending upon the implementation, a succession of **H** transforms can transform a matrix to an upper-triangular, lower-triangular, tridiagonal or other useful form. In practice, the Householder transform can be implemented in a highly efficient manner.

and $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$, so this reduces to

$$\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{x} = \mathbf{R}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{b}. \tag{A1.61}$$

Multiplying through by $\mathbf{R}^{-\mathrm{T}}$ (i.e., the inverse of $\mathbf{R}^{\mathrm{T}}$) gives

$$\mathbf{R}\mathbf{x} = \mathbf{Q}^{\mathrm{T}}\mathbf{b}, \tag{A1.62}$$

which is identical to (A1.54). In fact, *the solution of an overdetermined system Ax = b by QR decomposition leads directly to the least-squares solution*; (A1.58) is directly equivalent to the QR decomposition form of (A1.62).

Since **R** replaces $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and is upper triangular (i.e., solved by backsubstitution), we avoid computing $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$, which exacerbates problems of ill-conditioning. Orthogonal decomposition does not actually eliminate ill-conditioning – rather, by not computing the $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$, it reduces the sensitivity by reducing the loss of precision due to round-off error. In general, *orthogonal decomposition preserves about twice as many significant digits as does the comparable elimination (inverse) solution*, though the orthogonal decomposition requires more operations. Therefore, when a matrix inverse can be stably inverted by using double precision, an orthogonal decomposition will likely achieve the same level of stability using single precision, making "more" operations warranted.

If QR decomposition is used to solve weighted least-squares problems (done by modifying **A** and **b** prior to decomposition as discussed in Chapter 6), some row interchanges may be required if disparately weighted rows result. This may be necessary to avoid subtracting very small numbers from very large numbers, which can suffer from round-off errors during the backsubstitution.

A1.9 Statistical Topics

A1.9.1 Random Vectors and Matrices

We now consider the cases where we are working with vectors and matrices that are filled with random variables. Such vectors or matrices are themselves random vectors or matrices, since the actual elements occupying them can be any of a series of values representing the spread in each element. So, consider a random vector, $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix}, \tag{A1.63}$$

where the elements each represent a random variable, X_i. Depending on which values are present in the vector, any one particular vector represents just one realization of an infinite (or finite, if each random variable has a finite number of possible values or events) number of possible vectors, $\mathbf{X}$. Thus, the random vector is analogous to a scalar random variable.

Now consider the random matrix $\mathbf{Z}$:

$$\mathbf{Z} = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1m} \\ Z_{21} & Z_{22} & \cdots & Z_{2m} \\ Z_{31} & Z_{32} & \cdots & Z_{3m} \\ & & \vdots & \\ Z_{n1} & Z_{n2} & \cdots & Z_{nm} \end{bmatrix}, \tag{A1.64}$$

where again each element represents a random variable.

Random vectors and matrices represent an ideal means for manipulating multivariate random variables or situations in which we are dealing with systems of random variables.

A1.9.2 Statistical Moments of Random Matrices (Expectance Operations)

Mean

Now consider the expectance of a random matrix. As before, this is denoted E[**Z**], indicating

$$\begin{aligned} \mathrm{E}[Z] &= \begin{bmatrix} \mathrm{E}[Z_{11}] & \mathrm{E}[Z_{12}] & \cdots & \mathrm{E}[Z_{1m}] \\ \mathrm{E}[Z_{21}] & \mathrm{E}[Z_{22}] & \cdots & \mathrm{E}[Z_{2m}] \\ \mathrm{E}[Z_{31}] & \mathrm{E}[Z_{32}] & \cdots & \mathrm{E}[Z_{3m}] \\ & & \vdots & \\ \mathrm{E}[Z_{n1}] & \mathrm{E}[Z_{n2}] & \cdots & \mathrm{E}[Z_{nm}] \end{bmatrix} \\ &= \begin{bmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1m} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2m} \\ \mu_{31} & \mu_{32} & \cdots & \mu_{3m} \\ & & \vdots & \\ \mu_{n1} & \mu_{n2} & \cdots & \mu_{nm} \end{bmatrix}, \end{aligned} \tag{A1.65a}$$

or

$$= \boldsymbol{\mu}. \tag{A1.65b}$$

So, the expectance of random matrix $\mathbf{Z}$ is the mean, or mean matrix, $\boldsymbol{\mu}$. Likewise, for the random vector, $E[\mathbf{X}] = \boldsymbol{\mu}$. The elements of $\boldsymbol{\mu}$ contain the expected value of each element of $\mathbf{Z}$ or $\mathbf{X}$.

As shown previously, a scalar times a matrix is equal to the matrix with each individual element multiplied by the scalar. From this and (A1.65a), it is clear that

$$E[a\mathbf{X}] = aE[\mathbf{X}] = a\boldsymbol{\mu}. \tag{A1.66}$$

$$E[a\mathbf{X} + b] = aE[\mathbf{X}] + E[b] = a\boldsymbol{\mu} + b. \tag{A1.67}$$

Likewise for the vector products and sums, where **a, b** and **c** are nonrandom variable vectors, in which case,

$$E[\mathbf{aX}] = \mathbf{a}E[\mathbf{X}] = \mathbf{a}\boldsymbol{\mu}. \tag{A1.68}$$

$$E[\mathbf{aX} + \mathbf{b}] = \mathbf{a}E[\mathbf{X}] + E[\mathbf{b}] = \mathbf{a}\boldsymbol{\mu} + \mathbf{b}. \tag{A1.69}$$

$$E[\mathbf{aXb} + \mathbf{c}] = \mathbf{a}E[\mathbf{X}]\mathbf{b} + E[\mathbf{c}] = \mathbf{a}\boldsymbol{\mu}\mathbf{b} + \mathbf{c}. \tag{A1.70}$$

Variance (Covariance Matrix)

The variance of random vector, $\mathbf{X}$, is analogous to the variance of a scalar random variable, X, where it was then the expectance of a function of the random variable $(X - \mu)^2$, and in this case it is given as the expectance of a random matrix $\mathbf{Z}$ of the form

$$\mathbf{Z} = (\mathbf{X} - E[\mathbf{X}])(\mathbf{X} - E[\mathbf{X}])^T, \tag{A1.71}$$

so

$$\begin{aligned} \mathrm{Var}[\mathbf{X}] = \sum = E[\mathbf{Z}] &= E[(\mathbf{X} - E[\mathbf{X}])(\mathbf{X} - E[\mathbf{X}])^T] \\ &= E[(X - \mu)(X - \mu)^T], \end{aligned} \tag{A1.72a}$$

where $\boldsymbol{\Sigma}$ is the **covariance matrix** of $\mathbf{X}$ (some people refer to $\boldsymbol{\Sigma}$ as the **variance-covariance matrix**).

Equation (A1.72a) is expanded and reduced from (A1.68) as

$$\begin{aligned} &= E[(XX^T - \mu X^T - X\mu^T + \mu\mu^T] \\ &= E[XX^T] - E[\mu X^T] - E[X\mu^T] + E[\mu\mu^T] \\ &= E[XX^T] - \mu E[X^T] - E[X]\mu^T + \mu\mu^T \\ &= E[XX^T] - \mu\mu^T - \mu\mu^T + \mu\mu^T \\ &= E[XX^T] - \mu\mu^T. \end{aligned} \tag{A1.72b}$$

This is equivalent to the scalar operation

$$= E[(X_i - \mu_i)(X_j - \mu_j)] \tag{A1.72c}$$

for each element i,j, i = 1,2,3, …,n, j = 1,2,3, …,m. As a scalar operation for pairs of elements, it is clear (by considering the operation in (A1.72c) from Chapter 2 on Probability) that when $i \neq j$, this gives the covariance between the various pairs of random variables in $\mathbf{X}$, and when i = j, it is giving the variance.

Consider the vector operation in (A1.72a). This involves the *outer product* $(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^{\mathrm{T}}$. First, for the case where the mean vector is the null vector, i.e., the X random variables have zero mean, then this outer product is given as $\mathbf{X}\mathbf{X}^{\mathrm{T}}$, or

$$\mathbf{X}\mathbf{X}^{\mathrm{T}} = \begin{bmatrix} \mathrm{X}_1 \\ \mathrm{X}_2 \\ \mathrm{X}_3 \\ \dots \\ \mathrm{X}_{\mathrm{n}} \end{bmatrix} \begin{bmatrix} \mathrm{X}_1 & \mathrm{X}_2 & \dots & \mathrm{X}_n \end{bmatrix} = \begin{bmatrix} \mathrm{X}_1\mathrm{X}_1 & \mathrm{X}_1\mathrm{X}_2 & \dots & \mathrm{X}_1\mathrm{X}_{\mathrm{n}} \\ \mathrm{X}_2\mathrm{X}_1 & \mathrm{X}_2\mathrm{X}_2 & \dots & \mathrm{X}_2\mathrm{X}_{\mathrm{n}} \\ \mathrm{X}_3\mathrm{X}_1 & \mathrm{X}_3\mathrm{X}_2 & \dots & \mathrm{X}_3\mathrm{X}_{\mathrm{n}} \\ & & \vdots & \\ \mathrm{X}_{\mathrm{n}}\mathrm{X}_1 & \mathrm{X}_{\mathrm{n}}\mathrm{X}_2 & \dots & \mathrm{X}_{\mathrm{n}}\mathrm{X}_{\mathrm{n}} \end{bmatrix}, \tag{A1.73a}$$

and the expectance of this matrix, as indicated by (A1.65), is

$$\mathrm{E}[\mathbf{X}\mathbf{X}^{\mathrm{T}}] = \begin{bmatrix} \mathrm{E}[\mathrm{x}_1\mathrm{x}_1] & \mathrm{E}[\mathrm{x}_1\mathrm{x}_2] & \dots & \mathrm{E}[\mathrm{x}_1\mathrm{x}_{\mathrm{n}}] \\ \mathrm{E}[\mathrm{x}_2\mathrm{x}_1] & \mathrm{E}[\mathrm{x}_2\mathrm{x}_2] & \dots & \mathrm{E}[\mathrm{x}_2\mathrm{x}_{\mathrm{n}}] \\ \mathrm{E}[\mathrm{x}_3\mathrm{x}_1] & \mathrm{E}[\mathrm{x}_3\mathrm{x}_2] & \dots & \mathrm{E}[\mathrm{x}_3\mathrm{x}_{\mathrm{n}}] \\ & & \vdots & \\ \mathrm{E}[\mathrm{x}_{\mathrm{n}}\mathrm{x}_1] & \mathrm{E}[\mathrm{x}_{\mathrm{n}}\mathrm{x}_2] & \dots & \mathrm{E}[\mathrm{x}_{\mathrm{n}}\mathrm{x}_{\mathrm{n}}] \end{bmatrix}. \tag{A1.73b}$$

If the mean is not zero, then the operation is the same, except that the mean for each random variable is subtracted off so that the above matrix product looks like this:

$$\mathrm{E}[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^{\mathrm{T}}] = \begin{bmatrix} \mathrm{E}[(\mathrm{X}_1-\mu_1)(\mathrm{X}_1-\mu_1)] & \mathrm{E}[(\mathrm{X}_1-\mu_1)(\mathrm{X}_2-\mu_2)] & \dots & \mathrm{E}[(\mathrm{X}_1-\mu_1)(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})] \\ \mathrm{E}[(\mathrm{X}_2-\mu_2)(\mathrm{X}_1-\mu_1)] & \mathrm{E}[(\mathrm{X}_2-\mu_2)(\mathrm{X}_2-\mu_2)] & \dots & \mathrm{E}[(\mathrm{X}_2-\mu_2)(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})] \\ \mathrm{E}[(\mathrm{X}_3-\mu_3)(\mathrm{X}_1-\mu_1)] & \mathrm{E}[(\mathrm{X}_3-\mu_3)(\mathrm{X}_2-\mu_2)] & \dots & \mathrm{E}[(\mathrm{X}_3-\mu_3)(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})] \\ & & \vdots & \\ \mathrm{E}[(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})(\mathrm{X}_1-\mu_1)] & \mathrm{E}[(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})(\mathrm{X}_2-\mu_2)] & \dots & \mathrm{E}[(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})(\mathrm{X}_{\mathrm{n}}-\mu_{\mathrm{n}})] \end{bmatrix} \tag{A1.73c}$$

In either case, with $\boldsymbol{\mu}$ zero or not, when n equals the full population or ensemble size (N or ∞), (A1.73b,c) is

$$\mathrm{Var}[\mathbf{X}] = \Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1\mathrm{n}} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2\mathrm{n}} \\ \sigma_{31} & \sigma_{32} & \dots & \sigma_{3\mathrm{n}} \\ & & \vdots & \\ \sigma_{\mathrm{n}1} & \sigma_{\mathrm{n}2} & \dots & \sigma_{\mathrm{nn}} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1\mathrm{n}} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2\mathrm{n}} \\ \sigma_{31} & \sigma_{32} & \dots & \sigma_{3\mathrm{n}} \\ & & \vdots & \\ \sigma_{\mathrm{n}1} & \sigma_{\mathrm{n}2} & \dots & \sigma_{\mathrm{n}}^2 \end{bmatrix}, \tag{A1.73d}$$

where the $\sigma_{ii} = \sigma_i^2 =$ the true variance of the random variable $\mathbf{X}_i$ and the σ_{ij} = the true covariance between the random variable $\mathbf{X}_i$ and $\mathbf{X}_j$. When n is less than the full ensemble size, the above is the sample Var[**X**] = **S**, populated by sample variance (s_i^2) and covariance (s_{ij}) estimates, or lagged covariance estimates ($\gamma(k)$) for ergodic series.

Thus, the variance of random vector **X** is given by the covariance matrix, **Σ**.

Consider the variance of the following vector product and sum, where **a** and **b** are nonrandom variable vectors, in which case

$$\mathrm{Var}[\mathbf{aX}+\mathbf{b}] = \mathrm{E}[(\mathbf{aX}+\mathbf{b}-\mathrm{E}[\mathbf{aX}+\mathbf{b}])(\mathbf{aX}+\mathbf{b}-\mathrm{E}[\mathbf{aX}+\mathbf{b}])^{\mathrm{T}}]. \tag{A1.74a}$$

The expectance term $\mathrm{E}[aX+b] = a\mathrm{E}[X]+b = a\mu+b$ from (A1.69), so (A1.74a) becomes

$$\begin{aligned}
&=\mathrm{E}[(aX+b-\{a\mu+b\})(aX+b-\{a\mu+b\})^{\mathrm{T}}]\\
&=\mathrm{E}[(aX+b-a\mu-b)(aX+b-a\mu-b)^{\mathrm{T}}]\\
&=\mathrm{E}[(aX-a\mu)(aX-a\mu)^{\mathrm{T}}]\\
&=\mathrm{E}[(aXX^{\mathrm{T}}a^{\mathrm{T}}-a\mu X^{\mathrm{T}}a^{\mathrm{T}}-aX\mu^{\mathrm{T}}a^{\mathrm{T}}+a\mu\mu^{\mathrm{T}}a^{\mathrm{T}}]\\
&=\mathrm{E}[aXX^{\mathrm{T}}\,a^{\mathrm{T}}]-\mathrm{E}\,[a\mu X^{\mathrm{T}}a^{\mathrm{T}}]-\mathrm{E}[aX\mu^{\mathrm{T}}a^{\mathrm{T}}]+\mathrm{E}[a\mu\mu^{\mathrm{T}}a^{\mathrm{T}}]\\
&=\mathrm{E}[aXX^{\mathrm{T}}a^{\mathrm{T}}]-a\mu\mathrm{E}[X^{\mathrm{T}}]a^{\mathrm{T}}-a\mathrm{E}[X]\mu^{\mathrm{T}}a^{\mathrm{T}}+a\mu\mu^{\mathrm{T}}a^{\mathrm{T}}\\
&=a\mathrm{E}[XX^{\mathrm{T}}]a^{\mathrm{T}}-a\mu\mu^{\mathrm{T}}a^{\mathrm{T}}-a\mu\mu^{\mathrm{T}}a^{\mathrm{T}}+a\mu\mu^{\mathrm{T}}a^{\mathrm{T}}\\
&=a\mathrm{E}[XX^{\mathrm{T}}]a^{\mathrm{T}}-a\mu\mu^{\mathrm{T}}a^{\mathrm{T}},
\end{aligned} \tag{A1.74b}$$

which, from (A1.72b), is seen to be

$$\begin{aligned}
&=\mathbf{a}\mathrm{Var}[\mathbf{X}]\mathbf{a}^{\mathrm{T}}\\
&=\mathbf{a}\Sigma\mathbf{a}^{\mathrm{T}}.
\end{aligned} \tag{A1.74c}$$

The sample covariance matrix – that is, the estimate of **Σ** for random vector **X** – is done by estimating the elements of **Σ** using the standard formulas for sample variance, given in (3.6), and sample covariance, given in (3.7). These estimates are then placed in the covariance matrix as indicated in (A1.73).

For stationary and ergodic time series, the covariance matrix comes directly from the autocovariance function discussed in Chapter 7 (equation (7.43a)). For that case, all variances (in the principle diagonal) are equal, and the non-principle diagonal covariance terms are simply a function of lag k (so $\sigma_{21} = \sigma_{32} = \gamma(k=1)$), as defined in Chapter 7 and shown in (7.44).

A1.10 Matrix References

Davis, John C., 1973. *Statistics and Data Analysis in Geology*. John Wiley and Sons, New York, NY. Chapter 4.

Good overview of the relevant basic matrix algebra, though he skips many important concepts. Good for a short introduction to the subject, though.

Householder, Alston S., 1975. *Theory of Matrices in Numerical Analysis*. Dover Publications Inc., New York, NY. 257 pp.

In most cases, I have only listed books that I like – this is the exception. Since this is an inexpensive Dover book written by the man for which the Householder transform is named, it may seem like an obvious reference – wrong. This book is incomprehensible unless you have already mastered all of the material he is presenting (which he does in a very abstruse manner). Other than that, it's a great reference.

Jennings, Alan, 1977. *Matrix Computation for Engineers and Scientists*. John Wiley and Sons, New York, NY. 329 pp.

Good book for matrix algebra on computers.

Press, William H., Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, 1986. *Numerical Recipes*. Cambridge University Press, Cambridge. 818 pp.

Superb general computational text with relevant theory, practical implementation tips and subroutines for everything. Not a review of basic linear algebra, but good stuff for matrix inversions and decompositions of all types.

Searle, Shayle R., 1982. *Matrix Algebra Useful for Statistics*. John Wiley and Sons, New York, NY. 438 pp.

Excellent book with a lot of insights provided that are not found elsewhere. This is not a statistics book, but it does show numerous statistical applications in matrix form. Good discussion on derivatives of matrices.

Strang, Gilbert, 2003. *Introduction to Linear Algebra and Its Applications*, 3rd ed. Wellesley-Cambridge Press, Wellesley, MA. 568 pp.

Excellent introduction to linear algebra. I have the instructor's manual for questions (if you find yourself unable to answer a question), and even better, this book is the focus of Strang's introductory class (18.06). The videotaped lectures are available online at: web.mit.edu/18.06/www.

Strang, Gilbert, 1988. *Linear Algebra and Its Applications*, 3rd ed. Harcourt Brace Jovanovich, Publishers, San Diego, CA. 505 pp.

Great book for advanced linear algebra (one of my very favorites). This is not just an introduction to the technical manipulations of matrices; rather, Strang approaches it as an applied mathematician and gives insights and reasons for the various (often, seemingly esoteric) aspects.