Chapter 6

Matrices

In this chapter we introduce matrices and some basic operations on them. We give some applications in which they arise.

6.1 Matrices

A *matrix* is a rectangular array of numbers written between rectangular brackets, as in

$$\begin{bmatrix} 0 & 1 & -2.3 & 0.1 \\ 1.3 & 4 & -0.1 & 0 \\ 4.1 & -1 & 0 & 1.7 \end{bmatrix}.$$

It is also common to use large parentheses instead of rectangular brackets, as in

$$\begin{pmatrix} 0 & 1 & -2.3 & 0.1 \\ 1.3 & 4 & -0.1 & 0 \\ 4.1 & -1 & 0 & 1.7 \end{pmatrix}.$$

An important attribute of a matrix is its *size* or *dimensions*, *i.e.*, the numbers of rows and columns. The matrix above has 3 rows and 4 columns, so its size is 3×4. A matrix of size $m \times n$ is called an $m \times n$ matrix.

The *elements* (or *entries* or *coefficients*) of a matrix are the values in the array. The i, j element is the value in the ith row and jth column, denoted by double subscripts: the i, j element of a matrix A is denoted A_{ij} (or $A_{i,j}$, when i or j is more than one digit or character). The positive integers i and j are called the (row and column) *indices*. If A is an $m \times n$ matrix, then the row index i runs from 1 to m and the column index j runs from 1 to n. Row indices go from top to bottom, so row 1 is the top row and row m is the bottom row. Column indices go from left to right, so column 1 is the left column and column n is the right column.

If the matrix above is B, then we have $B_{13} = -2.3$, $B_{32} = -1$. The row index of the bottom left element (which has value 4.1) is 3; its column index is 1.

Two matrices are equal if they have the same size, and the corresponding entries are all equal. As with vectors, we normally deal with matrices with entries that

are real numbers, which will be our assumption unless we state otherwise. The set of real $m \times n$ matrices is denoted $\mathbf{R}^{m \times n}$. But matrices with complex entries, for example, do arise in some applications.

Matrix indexing. As with vectors, standard mathematical notation indexes the rows and columns of a matrix starting from 1. In computer languages, matrices are often (but not always) stored as 2-dimensional arrays, which can be indexed in a variety of ways, depending on the language. Lower level languages typically use indices starting from 0; higher level languages and packages that support matrix operations usually use standard mathematical indexing, starting from 1.

Square, tall, and wide matrices. A *square* matrix has an equal number of rows and columns. A square matrix of size $n \times n$ is said to be of *order* n. A *tall* matrix has more rows than columns (size $m \times n$ with $m > n$). A *wide* matrix has more columns than rows (size $m \times n$ with $n > m$).

Column and row vectors. An n-vector can be interpreted as an $n \times 1$ matrix; we do not distinguish between vectors and matrices with one column. A matrix with only one row, *i.e.*, with size $1 \times n$, is called a *row vector*; to give its size, we can refer to it as an *n-row-vector*. As an example,

$$\begin{bmatrix} -2.1 & -3 & 0 \end{bmatrix}$$

is a 3-row-vector (or 1×3 matrix). To distinguish them from row vectors, vectors are sometimes called *column vectors*. A 1×1 matrix is considered to be the same as a scalar.

Notational conventions. Many authors (including us) tend to use capital letters to denote matrices, and lower case letters for (column or row) vectors. But this convention is not standardized, so you should be prepared to figure out whether a symbol represents a matrix, column vector, row vector, or a scalar, from context. (The more considerate authors will tell you what the symbols represent, for example, by referring to 'the matrix A' when introducing it.)

Columns and rows of a matrix. An $m \times n$ matrix A has n columns, given by (the m-vectors)

$$a_j = \begin{bmatrix} A_{1j} \\ \vdots \\ A_{mj} \end{bmatrix},$$

for $j = 1, \ldots, n$. The same matrix has m rows, given by the (n-row-vectors)

$$b_i = \begin{bmatrix} A_{i1} & \cdots & A_{in} \end{bmatrix},$$

for $i = 1, \ldots, m$.

As a specific example, the 2×3 matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

has first row

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

(which is a 3-row-vector or a 1×3 matrix), and second column

$$\begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

(which is a 2-vector or 2×1 matrix), also written compactly as $(2, 5)$.

Block matrices and submatrices. It is useful to consider matrices whose entries are themselves matrices, as in

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix},$$

where B, C, D, and E are matrices. Such matrices are called *block matrices*; the elements B, C, D, and E are called *blocks* or *submatrices* of A. The submatrices can be referred to by their block row and column indices; for example, C is the 1,2 block of A.

Block matrices must have the right dimensions to fit together. Matrices in the same (block) row must have the same number of rows (*i.e.*, the same 'height'); matrices in the same (block) column must have the same number of columns (*i.e.*, the same 'width'). In the example above, B and C must have the same number of rows, and C and E must have the same number of columns. Matrix blocks placed next to each other in the same row are said to be *concatenated*; matrix blocks placed above each other are called *stacked*.

As an example, consider

$$B = \begin{bmatrix} 0 & 2 & 3 \end{bmatrix}, \qquad C = \begin{bmatrix} -1 \end{bmatrix}, \qquad D = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & 5 \end{bmatrix}, \qquad E = \begin{bmatrix} 4 \\ 4 \end{bmatrix}.$$

Then the block matrix A above is given by

$$A = \begin{bmatrix} 0 & 2 & 3 & -1 \\ 2 & 2 & 1 & 4 \\ 1 & 3 & 5 & 4 \end{bmatrix}. \tag{6.1}$$

(Note that we have dropped the left and right brackets that delimit the blocks. This is similar to the way we drop the brackets in a 1×1 matrix to get a scalar.)

We can also divide a larger matrix (or vector) into 'blocks'. In this context the blocks are called *submatrices* of the big matrix. As with vectors, we can use colon notation to denote submatrices. If A is an $m \times n$ matrix, and p, q, r, s are integers with $1 \le p \le q \le m$ and $1 \le r \le s \le n$, then $A_{p:q,r:s}$ denotes the submatrix

$$A_{p:q,r:s} = \begin{bmatrix} A_{pr} & A_{p,r+1} & \cdots & A_{ps} \\ A_{p+1,r} & A_{p+1,r+1} & \cdots & A_{p+1,s} \\ \vdots & \vdots & & \vdots \\ A_{qr} & A_{q,r+1} & \cdots & A_{qs} \end{bmatrix}.$$

This submatrix has size $(q-p+1) \times (s-r+1)$ and is obtained by extracting from A the elements in rows p through q and columns r through s.

For the specific matrix A in (6.1), we have

$$A_{2:3,3:4} = \begin{bmatrix} 1 & 4 \\ 5 & 4 \end{bmatrix}.$$

Column and row representation of a matrix. Using block matrix notation we can write an $m \times n$ matrix A as a block matrix with one block row and n block columns,

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix},$$

where a_j, which is an m-vector, is the jth column of A. Thus, an $m \times n$ matrix can be viewed as its n columns, concatenated.

Similarly, an $m \times n$ matrix A can be written as a block matrix with one block column and m block rows:

$$A = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

where b_i, which is a row n-vector, is the ith row of A. In this notation, the matrix A is interpreted as its m rows, stacked.

Examples

Table interpretation. The most direct interpretation of a matrix is as a table of numbers that depend on two indices, i and j. (A vector is a list of numbers that depend on only one index.) In this case the rows and columns of the matrix usually have some simple interpretation. Some examples are given below.

- *Images.* A black and white image with $M \times N$ pixels is naturally represented as an $M \times N$ matrix. The row index i gives the vertical position of the pixel, the column index j gives the horizontal position of the pixel, and the i, j entry gives the pixel value.

- *Rainfall data.* An $m \times n$ matrix A gives the rainfall at m different locations on n consecutive days, so A_{42} (which is a number) is the rainfall at location 4 on day 2. The jth column of A, which is an m-vector, gives the rainfall at the m locations on day j. The ith row of A, which is an n-row-vector, is the time series of rainfall at location i.

- *Asset returns.* A $T \times n$ matrix R gives the returns of a collection of n assets (called the *universe* of assets) over T periods, with R_{ij} giving the return of asset j in period i. So $R_{12,7} = -0.03$ means that asset 7 had a 3% loss in period 12. The 4th column of R is a T-vector that is the return time series

Date	AAPL	GOOG	MMM	AMZN
March 1, 2016	0.00219	0.00006	−0.00113	0.00202
March 2, 2016	0.00744	−0.00894	−0.00019	−0.00468
March 3, 2016	0.01488	−0.00215	0.00433	−0.00407

Table 6.1 Daily returns of Apple (AAPL), Google (GOOG), 3M (MMM), and Amazon (AMZN), on March 1, 2, and 3, 2016 (based on closing prices).

for asset 4. The 3rd row of R is an n-row-vector that gives the returns of all assets in the universe in period 3.

An example of an asset return matrix, with a universe of $n = 4$ assets over $T = 3$ periods, is shown in table 6.1.

- *Prices from multiple suppliers.* An $m \times n$ matrix P gives the prices of n different goods from m different suppliers (or locations): P_{ij} is the price that supplier i charges for good j. The jth column of P is the m-vector of supplier prices for good j; the ith row gives the prices for all goods from supplier i.

- *Contingency table.* Suppose we have a collection of objects with two attributes, the first attribute with m possible values and the second with n possible values. An $m \times n$ matrix A can be used to hold the counts of the numbers of objects with the different pairs of attributes: A_{ij} is the number of objects with first attribute i and second attribute j. (This is the analog of a count n-vector, that records the counts of one attribute in a collection.) For example, a population of college students can be described by a 4×50 matrix, with the i, j entry the number of students in year i of their studies, from state j (with the states ordered in, say, alphabetical order). The ith row of A gives the geographic distribution of students in year i of their studies; the jth column of A is a 4-vector giving the numbers of student from state j in their first through fourth years of study.

- *Customer purchase history.* An $n \times N$ matrix P can be used to store a set of N customers' purchase histories of n products, items, or services, over some period. The entry P_{ij} represents the dollar value of product i that customer j purchased over the period (or as an alternative, the number or quantity of the product). The jth column of P is the purchase history vector for customer j; the ith row gives the sales report for product i across the N customers.

Matrix representation of a collection of vectors. Matrices are very often used as a compact way to give a set of indexed vectors of the same size. For example, if $x_1, \ldots, x_N$ are n-vectors that give the n feature values for each of N objects, we can collect them all into one $n \times N$ matrix

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix},$$

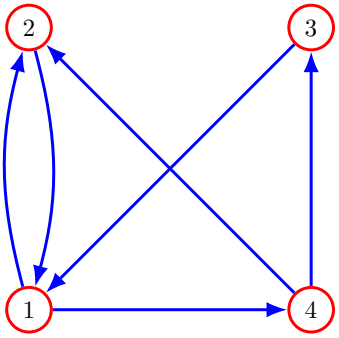

Figure 6.1 The relation (6.2) as a directed graph.

often called a *data matrix* or *feature matrix*. Its jth column is the feature n-vector for the jth object (in this context sometimes called the jth *example*). The ith row of the data matrix X is an N-row-vector whose entries are the values of the ith feature across the examples. We can also directly interpret the entries of the data matrix: X_{ij} (which is a number) is the value of the ith feature for the jth example.

As another example, a $3 \times M$ matrix can be used to represent a collection of M locations or positions in 3-D space, with its jth column giving the jth position.

Matrix representation of a relation or graph. Suppose we have n objects labeled $1, \ldots, n$. A *relation* $\mathcal{R}$ on the set of objects $\{1, \ldots, n\}$ is a subset of ordered pairs of objects. As an example, $\mathcal{R}$ can represent a *preference relation* among n possible products or choices, with $(i, j) \in \mathcal{R}$ meaning that choice i is preferred to choice j.

A relation can also be viewed as a *directed graph*, with nodes (or vertices) labeled $1, \ldots, n$, and a directed edge from j to i for each $(i, j) \in \mathcal{R}$. This is typically drawn as a graph, with arrows indicating the direction of the edge, as shown in figure 6.1, for the relation on 4 objects

$$\mathcal{R} = \{(1,2),\ (1,3),\ (2,1),\ (2,4),\ (3,4),\ (4,1)\}. \tag{6.2}$$

A relation $\mathcal{R}$ on $\{1, \ldots, n\}$ is represented by the $n \times n$ matrix A with

$$A_{ij} = \begin{cases} 1 & (i,j) \in \mathcal{R} \\ 0 & (i,j) \notin \mathcal{R}. \end{cases}$$

This matrix is called the *adjacency matrix* associated with the graph. (Some authors define the adjacency matrix in the reverse sense, with $A_{ij} = 1$ meaning there is an edge from i to j.) The relation (6.2), for example, is represented by the matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

This is the adjacency matrix of the associated graph, shown in figure 6.1. (We will encounter another matrix associated with a directed graph in §7.3.)

6.2 Zero and identity matrices

Zero matrix. A zero matrix is a matrix with all elements equal to zero. The zero matrix of size $m \times n$ is sometimes written as $0_{m\times n}$, but usually a zero matrix is denoted just 0, the same symbol used to denote the number 0 or zero vectors. In this case the size of the zero matrix must be determined from the context.

Identity matrix. An identity matrix is another common matrix. It is always square. Its *diagonal* elements, *i.e.*, those with equal row and column indices, are all equal to one, and its off-diagonal elements (those with unequal row and column indices) are zero. Identity matrices are denoted by the letter I. Formally, the identity matrix of size n is defined by

$$I_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j, \end{cases}$$

for $i, j = 1, \ldots, n$. For example,

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

are the 2×2 and 4×4 identity matrices.

The column vectors of the $n \times n$ identity matrix are the unit vectors of size n. Using block matrix notation, we can write

$$I = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix},$$

where e_k is the kth unit vector of size n.

Sometimes a subscript is used to denote the size of an identity matrix, as in I_4 or $I_{2\times 2}$. But more often the size is omitted and follows from the context. For example, if

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

then

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

The dimensions of the two identity matrices follow from the size of A. The identity matrix in the 1,1 position must be 2×2, and the identity matrix in the 2,2 position must be 3×3. This also determines the size of the zero matrix in the 2,1 position.

The importance of the identity matrix will become clear later, in §10.1.

Sparse matrices. A matrix A is said to be *sparse* if many of its entries are zero, or (put another way) just a few of its entries are nonzero. Its *sparsity pattern* is the set of indices (i, j) for which $A_{ij} \neq 0$. The *number of nonzeros* of a sparse matrix A is the number of entries in its sparsity pattern, and denoted $\mathbf{nnz}(A)$. If A is $m \times n$ we have $\mathbf{nnz}(A) \leq mn$. Its *density* is $\mathbf{nnz}(A)/(mn)$, which is no more than one. Densities of sparse matrices that arise in applications are typically small or very small, as in 10^{-2} or 10^{-4}. There is no precise definition of how small the density must be for a matrix to qualify as sparse. A famous definition of sparse matrix due to the mathematician James H. Wilkinson is: A matrix is sparse if it has enough zero entries that it pays to take advantage of them. Sparse matrices can be stored and manipulated efficiently on a computer.

Many common matrices are sparse. An $n \times n$ identity matrix is sparse, since it has only n nonzeros, so its density is $1/n$. The zero matrix is the sparsest possible matrix, since it has no nonzero entries. Several special sparsity patterns have names; we describe some important ones below.

Like sparse vectors, sparse matrices arise in many applications. A typical customer purchase history matrix (see page 111) is sparse, since each customer has likely only purchased a small fraction of all the products available.

Diagonal matrices. A square $n \times n$ matrix A is *diagonal* if $A_{ij} = 0$ for $i \neq j$. (The entries of a matrix with $i = j$ are called the *diagonal entries*; those with $i \neq j$ are its *off-diagonal* entries.) A diagonal matrix is one for which all off-diagonal entries are zero. Examples of diagonal matrices we have already seen are square zero matrices and identity matrices. Other examples are

$$\begin{bmatrix} -3 & 0 \\ 0 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 1.2 \end{bmatrix}.$$

(Note that in the first example, one of the diagonal elements is also zero.)

The notation $\mathbf{diag}(a_1, \ldots, a_n)$ is used to compactly describe the $n \times n$ diagonal matrix A with diagonal entries $A_{11} = a_1, \ldots, A_{nn} = a_n$. This notation is not yet standard, but is coming into more prevalent use. As examples, the matrices above would be expressed as

$$\mathbf{diag}(-3, 0), \qquad \mathbf{diag}(0.2, -3, 1.2),$$

respectively. We also allow **diag** to take one n-vector argument, as in $I = \mathbf{diag}(\mathbf{1})$.

Triangular matrices. A square $n \times n$ matrix A is *upper triangular* if $A_{ij} = 0$ for $i > j$, and it is *lower triangular* if $A_{ij} = 0$ for $i < j$. (So a diagonal matrix is one that is both lower and upper triangular.) If a matrix is either lower or upper triangular, it is called *triangular*. For example, the matrices

$$\begin{bmatrix} 1 & -1 & 0.7 \\ 0 & 1.2 & -1.1 \\ 0 & 0 & 3.2 \end{bmatrix}, \qquad \begin{bmatrix} -0.6 & 0 \\ -0.3 & 3.5 \end{bmatrix},$$

are upper and lower triangular, respectively.

A triangular $n \times n$ matrix A has up to $n(n+1)/2$ nonzero entries, *i.e.*, around half its entries are zero. Triangular matrices are generally not considered sparse matrices, since their density is around 50%, but their special sparsity pattern will be important in the sequel.

6.3 Transpose, addition, and norm

6.3.1 Matrix transpose

If A is an $m \times n$ matrix, its *transpose*, denoted A^T (or sometimes A' or A^*), is the $n \times m$ matrix given by $(A^T)_{ij} = A_{ji}$. In words, the rows and columns of A are transposed in A^T. For example,

$$\begin{bmatrix} 0 & 4 \\ 7 & 0 \\ 3 & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 7 & 3 \\ 4 & 0 & 1 \end{bmatrix}.$$

If we transpose a matrix twice, we get back the original matrix: $(A^T)^T = A$. (The superscript T in the transpose is the same one used to denote the inner product of two n-vectors; we will soon see how they are related.)

Row and column vectors. Transposition converts row vectors into column vectors and vice versa. It is sometimes convenient to express a row vector as a^T, where a is a column vector. For example, we might refer to the m rows of an $m \times n$ matrix A as $\tilde{a}_i^T, \ldots, \tilde{a}_m^T$, where $\tilde{a}_1, \ldots, \tilde{a}_m$ are (column) n-vectors. As an example, the second row of the matrix

$$\begin{bmatrix} 0 & 7 & 3 \\ 4 & 0 & 1 \end{bmatrix}$$

can be written as (the row vector) $(4, 0, 1)^T$.

It is common to extend concepts from (column) vectors to row vectors, by applying the concept to the transposed row vectors. We say that a collection of row vectors is linearly dependent (or independent) if their transposes (which are column vectors) are linearly dependent (or independent). For example, 'the rows of a matrix A are linearly independent' means that the columns of A^T are linearly independent. As another example, 'the rows of a matrix A are orthonormal' means that their transposes, the columns of A^T, are orthonormal. 'Clustering the rows of a matrix X' means clustering the columns of X^T.

Transpose of block matrix. The transpose of a block matrix has the simple form (shown here for a 2×2 block matrix)

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^T = \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix},$$

where A, B, C, and D are matrices with compatible sizes. The transpose of a block matrix is the transposed block matrix, with each element transposed.

Document-term matrix. Consider a corpus (collection) of N documents, with word count vectors for a dictionary with n words. The *document-term* matrix associated with the corpus is the $N \times n$ matrix A, with A_{ij} the number of times word j appears in document i. The rows of the document-term matrix are $a_1^T, \ldots, a_N^T$, where the n-vectors $a_1, \ldots, a_N$ are the word count vectors for documents $1, \ldots, N$, respectively. The columns of the document-term matrix are also interesting. The jth column of A, which is an N-vector, gives the number of times word j appears in the corpus of N documents.

Data matrix. A collection of N n-vectors, for example feature n-vectors associated with N objects, can be given as an $n \times N$ matrix whose N columns are the vectors, as described on page 111. It is also common to describe this collection of vectors using the transpose of that matrix. In this case, we give the vectors as an $N \times n$ matrix X. Its ith row is x_i^T, the transpose of the ith vector. Its jth column gives the value of the jth entry (or feature) across the collection of N vectors. When an author refers to a *data matrix* or *feature matrix*, it can usually be determined from context (for example, its dimensions) whether they mean the data matrix organized by rows or columns.

Symmetric matrix. A square matrix A is *symmetric* if $A = A^T$, *i.e.*, $A_{ij} = A_{ji}$ for all i, j. Symmetric matrices arise in several applications. For example, suppose that A is the adjacency matrix of a graph or relation (see page 112). The matrix A is symmetric when the relation is symmetric, *i.e.*, whenever $(i, j) \in \mathcal{R}$, we also have $(j, i) \in \mathcal{R}$. An example is the *friend relation* on a set of n people, where $(i, j) \in \mathcal{R}$ means that person i and person j are friends. (In this case the associated graph is called the 'social network graph'.)

6.3.2 Matrix addition

Two matrices of the same size can be added together. The result is another matrix of the same size, obtained by adding the corresponding elements of the two matrices. For example,

$$\begin{bmatrix} 0 & 4 \\ 7 & 0 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 6 \\ 9 & 3 \\ 3 & 5 \end{bmatrix}.$$

Matrix subtraction is similar. As an example,

$$\begin{bmatrix} 1 & 6 \\ 9 & 3 \end{bmatrix} - I = \begin{bmatrix} 0 & 6 \\ 9 & 2 \end{bmatrix}.$$

(This gives another example where we have to figure out the size of the identity matrix. Since we can only add or subtract matrices of the same size, I refers to a 2×2 identity matrix.)

Properties of matrix addition. The following important properties of matrix addition can be verified directly from the definition. We assume here that A, B, and C are matrices of the same size.

- *Commutativity.* $A + B = B + A$.
- *Associativity.* $(A + B) + C = A + (B + C)$. We therefore write both as $A + B + C$.
- *Addition with zero matrix.* $A + 0 = 0 + A = A$. Adding the zero matrix to a matrix has no effect.
- *Transpose of sum.* $(A + B)^T = A^T + B^T$. The transpose of a sum of two matrices is the sum of their transposes.

6.3.3 Scalar-matrix multiplication

Scalar multiplication of matrices is defined in a similar way as for vectors, and is done by multiplying every element of the matrix by the scalar. For example

$$(-2)\begin{bmatrix} 1 & 6 \\ 9 & 3 \\ 6 & 0 \end{bmatrix} = \begin{bmatrix} -2 & -12 \\ -18 & -6 \\ -12 & 0 \end{bmatrix}.$$

As with scalar-vector multiplication, the scalar can also appear on the right. Note that $0\,A = 0$ (where the left-hand zero is the scalar zero, and the right-hand 0 is the zero matrix).

Several useful properties of scalar multiplication follow directly from the definition. For example, $(\beta A)^T = \beta(A^T)$ for a scalar β and a matrix A. If A is a matrix and β, γ are scalars, then

$$(\beta + \gamma)A = \beta A + \gamma A, \qquad (\beta\gamma)A = \beta(\gamma A).$$

It is useful to identify the symbols appearing in these two equations. The $+$ symbol on the left of the left-hand equation is addition of scalars, while the $+$ symbol on the right of the left-hand equation denotes matrix addition. On the left side of the right-hand equation we see scalar-scalar multiplication ($\alpha\beta$) and scalar-matrix multiplication; on the right we see two cases of scalar-matrix multiplication.

Finally, we mention that scalar-matrix multiplication has higher precedence than matrix addition, which means that we should carry out multiplication before addition (when there are no parentheses to fix the order). So the right-hand side of the left equation above is to be interpreted as $(\beta A) + (\gamma A)$.

6.3.4 Matrix norm

The norm of an $m \times n$ matrix A, denoted $\|A\|$, is the squareroot of the sum of the squares of its entries,

$$\|A\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}. \tag{6.3}$$

This agrees with our definition for vectors when A is a vector, *i.e.*, $n = 1$. The norm of an $m \times n$ matrix is the norm of an mn-vector formed from the entries of the matrix (in any order). Like the vector norm, the matrix norm is a quantitative measure of the magnitude of a matrix. In some applications it is more natural to use the RMS values of the matrix entries, $\|A\|/\sqrt{mn}$, as a measure of matrix size. The RMS value of the matrix entries tells us the typical size of the entries, independent of the matrix dimensions.

The matrix norm (6.3) satisfies the properties of any norm, given on page 46. For any $m \times n$ matrix A, we have $\|A\| \geq 0$ (*i.e.*, the norm is nonnegative), and $\|A\| = 0$ only if $A = 0$ (definiteness). The matrix norm is nonnegative homogeneous: For any scalar γ and $m \times n$ matrix A, we have $\|\gamma A\| = |\gamma|\|A\|$. Finally, for any two $m \times n$ matrices A and B, we have the triangle inequality,

$$\|A + B\| \leq \|A\| + \|B\|.$$

(The plus symbol on the left-hand side is matrix addition, and the plus symbol on the right-hand side is addition of numbers.)

The matrix norm allows us to define the distance between two matrices as $\|A - B\|$. As with vectors, we can say that one matrix is close to, or near, another one if their distance is small. (What qualifies as small depends on the application.)

In this book we will only use the matrix norm (6.3). Several other norms of a matrix are commonly used, but are beyond the scope of this book. In contexts where other norms of a matrix are used, the norm (6.3) is called the *Frobenius norm*, after the mathematician Ferdinand Georg Frobenius, and is usually denoted with a subscript, as $\|A\|_F$.

One simple property of the matrix norm is $\|A\| = \|A^T\|$, *i.e.*, the norm of a matrix is the same as the norm of its transpose. Another one is

$$\|A\|^2 = \|a_1\|^2 + \cdots + \|a_n\|^2,$$

where $a_1, \ldots, a_n$ are the columns of A. In other words: The squared norm of a matrix is the sum of the squared norms of its columns.

6.4 Matrix-vector multiplication

If A is an $m \times n$ matrix and x is an n-vector, then the *matrix-vector product* $y = Ax$ is the m-vector y with elements

$$y_i = \sum_{k=1}^{n} A_{ik}x_k = A_{i1}x_1 + \cdots + A_{in}x_n, \quad i = 1, \ldots, m. \tag{6.4}$$

As a simple example, we have

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

Row and column interpretations. We can express the matrix-vector product in terms of the rows or columns of the matrix. From (6.4) we see that y_i is the inner product of x with the ith row of A:

$$y_i = b_i^T x, \quad i = 1, \ldots, m,$$

where b_i^T is the row i of A. The matrix-vector product can also be interpreted in terms of the columns of A. If a_k is the kth column of A, then $y = Ax$ can be written

$$y = x_1 a_1 + x_2 a_2 + \cdots + x_n a_n.$$

This shows that $y = Ax$ is a linear combination of the columns of A; the coefficients in the linear combination are the elements of x.

General examples. In the examples below, A is an $m \times n$ matrix and x is an n-vector.

- *Zero matrix.* When $A = 0$, we have $Ax = 0$. In other words, $0x = 0$. (The left-hand 0 is an $m \times n$ matrix, and the right-hand zero is an m-vector.)
- *Identity.* We have $Ix = x$ for any vector x. (The identity matrix here has dimension $n \times n$.) In other words, multiplying a vector by the identity matrix gives the same vector.
- *Picking out columns and rows.* An important identity is $Ae_j = a_j$, the jth column of A. Multiplying a unit vector by a matrix 'picks out' one of the columns of the matrix. $A^T e_i$, which is an n-vector, is the ith row of A, transposed. (In other words, $(A^T e_i)^T$ is the ith row of A.)
- *Summing or averaging columns or rows.* The m-vector $A\mathbf{1}$ is the sum of the columns of A; its ith entry is the sum of the entries in the ith row of A. The m-vector $A(\mathbf{1}/n)$ is the average of the columns of A; its ith entry is the average of the entries in the ith row of A. In a similar way, $A^T\mathbf{1}$ is an n-vector, whose jth entry is the sum of the entries in the jth column of A.
- *Difference matrix.* The $(n-1) \times n$ matrix

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & \ddots & \ddots & & & \\ & & & \ddots & \ddots & & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{6.5}$$

(where entries not shown are zero, and entries with diagonal dots are 1 or -1, continuing the pattern) is called the *difference matrix*. The vector Dx is the $(n-1)$-vector of differences of consecutive entries of x:

$$Dx = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \vdots \\ x_n - x_{n-1} \end{bmatrix}.$$

- *Running sum matrix.* The $n \times n$ matrix

$$S = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ 1 & 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \tag{6.6}$$

  is called the *running sum matrix*. The ith entry of the n-vector Sx is the sum of the first i entries of x:

$$Sx = \begin{bmatrix} x_1 \\ x_1 + x_2 \\ x_1 + x_2 + x_3 \\ \vdots \\ x_1 + \cdots + x_n \end{bmatrix}.$$

Application examples.

- *Feature matrix and weight vector.* Suppose X is a feature matrix, where its N columns $x_1, \ldots, x_N$ are feature n-vectors for N objects or examples. Let the n-vector w be a *weight vector*, and let $s_i = x_i^T w$ be the score associated with object i using the weight vector w. Then we can write $s = X^T w$, where s is the N-vector of scores of the objects.

- *Portfolio return time series.* Suppose that R is a $T \times n$ asset return matrix, that gives the returns of n assets over T periods. A common trading strategy maintains constant investment weights given by the n-vector w over the T periods. For example, $w_4 = 0.15$ means that 15% of the total portfolio value is held in asset 4. (Short positions are denoted by negative entries in w.) Then Rw, which is a T-vector, is the time series of the portfolio returns over the periods $1, \ldots, T$.

  As an example, consider a portfolio of the 4 assets in table 6.1, with weights $w = (0.4, 0.3, -0.2, 0.5)$. The product $Rw = (0.00213, -0.00201, 0.00241)$ gives the portfolio returns over the three periods in the example.

- *Polynomial evaluation at multiple points.* Suppose the entries of the n-vector c are the coefficients of a polynomial p of degree $n-1$ or less:

$$p(t) = c_1 + c_2 t + \cdots + c_{n-1} t^{n-2} + c_n t^{n-1}.$$

  Let $t_1, \ldots, t_m$ be m numbers, and define the m-vector y as $y_i = p(t_i)$. Then we have $y = Ac$, where A is the $m \times n$ matrix

$$A = \begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-2} & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-2} & t_2^{n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & t_m & \cdots & t_m^{n-2} & t_m^{n-1} \end{bmatrix}. \tag{6.7}$$

So multiplying a vector c by the matrix A is the same as evaluating a polynomial with coefficients c at m points. The matrix A in (6.7) comes up often, and is called a *Vandermonde matrix* (of degree $n-1$, at the points $t_1, \ldots, t_m$), named for the mathematician Alexandre-Théophile Vandermonde.

- *Total price from multiple suppliers.* Suppose the $m \times n$ matrix P gives the prices of n goods from m suppliers (or in m different locations). If q is an n-vector of quantities of the n goods (sometimes called a *basket* of goods), then $c = Pq$ is an N-vector that gives the total cost of the goods, from each of the N suppliers.

- *Document scoring.* Suppose A in an $N \times n$ document-term matrix, which gives the word counts of a corpus of N documents using a dictionary of n words, so the rows of A are the word count vectors for the documents. Suppose that w in an n-vector that gives a set of weights for the words in the dictionary. Then $s = Aw$ is an N-vector that gives the scores of the documents, using the weights and the word counts. A search engine, for example, might choose w (based on the search query) so that the scores are predictions of relevance of the documents (to the search).

- *Audio mixing.* Suppose the k columns of A are vectors representing audio signals or tracks of length T, and w is a k-vector. Then $b = Aw$ is a T-vector representing the mix of the audio signals, with track weights given by the vector w.

Inner product. When a and b are n-vectors, $a^T b$ is exactly the inner product of a and b, obtained from the rules for transposing matrices and forming a matrix-vector product. We start with the (column) n-vector a, consider it as an $n \times 1$ matrix, and transpose it to obtain the n-row-vector a^T. Now we multiply this $1 \times n$ matrix by the n-vector b, to obtain the 1-vector $a^T b$, which we also consider a scalar. So the notation $a^T b$ for the inner product is just a special case of matrix-vector multiplication.

Linear dependence of columns. We can express the concepts of linear dependence and independence in a compact form using matrix-vector multiplication. The columns of a matrix A are linearly dependent if $Ax = 0$ for some $x \neq 0$. The columns of a matrix A are linearly independent if $Ax = 0$ implies $x = 0$.

Expansion in a basis. If the columns of A are a basis, which means A is square with linearly independent columns $a_1, \ldots, a_n$, then for any n-vector b there is a unique n-vector x that satisfies $Ax = b$. In this case the vector x gives the coefficients in the expansion of b in the basis $a_1, \ldots, a_n$.

Properties of matrix-vector multiplication. Matrix-vector multiplication satisfies several properties that are readily verified. First, it distributes across the vector argument: For any $m \times n$ matrix A and any n-vectors u and v, we have

$$A(u + v) = Au + Av.$$

Matrix-vector multiplication, like ordinary multiplication of numbers, has higher precedence than addition, which means that when there are no parentheses to force the order of evaluation, multiplications are to be carried out before additions. This means that the right-hand side above is to be interpreted as $(Au) + (Av)$. The equation above looks innocent and natural, but must be read carefully. On the left-hand side, we first add the vectors u and v, which is the addition of n-vectors. We then multiply the resulting n-vector by the matrix A. On the right-hand side, we first multiply each of n-vectors by the matrix A (this is two matrix-vector multiplies); and then add the two resulting m-vectors together. The left- and right-hand sides of the equation above involve very different steps and operations, but the final result of each is the same m-vector.

Matrix-vector multiplication also distributes across the matrix argument: For any $m \times n$ matrices A and B, and any n-vector u, we have

$$(A + B)u = Au + Bu.$$

On the left-hand side the plus symbol is matrix addition; on the right-hand side it is vector addition.

Another basic property is, for any $m \times n$ matrix A, any n-vector u, and any scalar α, we have

$$(\alpha A)u = \alpha(Au)$$

(and so we can write this as αAu). On the left-hand side, we have scalar-matrix multiplication, followed by matrix-vector multiplication; on the right-hand side, we start with matrix-vector multiplication, and then perform scalar-vector multiplication. (Note that we also have $\alpha Au = A(\alpha u)$.)

Input-output interpretation. We can interpret the relation $y = Ax$, with A an $m \times n$ matrix, as a mapping from the n-vector x to the m-vector y. In this context we might think of x as an input, and y as the corresponding output. From equation (6.4), we can interpret A_{ij} as the factor by which y_i depends on x_j. Some examples of conclusions we can draw are given below.

- If A_{23} is positive and large, then y_2 depends strongly on x_3, and increases as x_3 increases.
- If A_{32} is much larger than the other entries in the third row of A, then y_3 depends much more on x_2 than the other inputs.
- If A is square and lower triangular, then y_i only depends on $x_1, \ldots, x_i$.

6.5 Complexity

Computer representation of matrices. An $m \times n$ matrix is usually represented on a computer as an $m \times n$ array of floating point numbers, which requires $8mn$ bytes. In some software systems symmetric matrices are represented in a more efficient way, by only storing the upper triangular elements in the matrix, in some

specific order. This reduces the memory requirement by around a factor of two. Sparse matrices are represented by various methods that encode for each nonzero element its row index i (an integer), its column index j (an integer) and its value A_{ij} (a floating point number). When the row and column indices are represented using 4 bytes (which allows m and n to range up to around 4.3 billion) this requires a total of around $16\,\mathbf{nnz}(A)$ bytes.

Complexity of matrix addition, scalar multiplication, and transposition. The addition of two $m \times n$ matrices or a scalar multiplication of an $m \times n$ matrix each take mn flops. When A is sparse, scalar multiplication requires $\mathbf{nnz}(A)$ flops. When at least one of A and B is sparse, computing $A + B$ requires at most $\min\{\mathbf{nnz}(A), \mathbf{nnz}(B)\}$ flops. (For any entry i, j for which one of A_{ij} or B_{ij} is zero, no arithmetic operations are needed to find $(A+B)_{ij}$.) Matrix transposition, *i.e.*, computing A^T, requires zero flops, since we simply copy entries of A to those of A^T. (Copying the entries does take time to carry out, but this is not reflected in the flop count.)

Complexity of matrix-vector multiplication. A matrix-vector multiplication of an $m \times n$ matrix A with an n-vector x requires $m(2n-1)$ flops, which we simplify to $2mn$ flops. This can be seen as follows. The result $y = Ax$ of the product is an m-vector, so there are m numbers to compute. The ith element of y is the inner product of the ith row of A and the vector x, which takes $2n-1$ flops.

If A is sparse, computing Ax requires $\mathbf{nnz}(A)$ multiplies (of A_{ij} and x_j, for each nonzero entry of A) and a number of additions that is no more than $\mathbf{nnz}(A)$. Thus, the complexity is between $\mathbf{nnz}(A)$ and $2\,\mathbf{nnz}(A)$ flops. As a special example, suppose A is $n \times n$ and diagonal. Then Ax can be computed with n multiplies (A_{ii} times x_i) and no additions, a total of $n = \mathbf{nnz}(A)$ flops.

Exercises

6.1 *Matrix and vector notation.* Suppose $a_1, \ldots, a_n$ are m-vectors. Determine whether each expression below makes sense (*i.e.*, uses valid notation). If the expression does make sense, give its dimensions.

(a) $\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$

(b) $\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$

(c) $\begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$

(d) $\begin{bmatrix} a_1^T & \cdots & a_n^T \end{bmatrix}$

6.2 *Matrix notation.* Suppose the block matrix

$$\begin{bmatrix} A & I \\ I & C \end{bmatrix}$$

makes sense, where A is a $p \times q$ matrix. What are the dimensions of C?

6.3 *Block matrix.* Assuming the matrix

$$K = \begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}$$

makes sense, which of the following statements must be true? ('Must be true' means that it follows with no additional assumptions.)

(a) K is square.

(b) A is square or wide.

(c) K is symmetric, *i.e.*, $K^T = K$.

(d) The identity and zero submatrices in K have the same dimensions.

(e) The zero submatrix is square.

6.4 *Adjacency matrix row and column sums.* Suppose A is the adjacency matrix of a directed graph (see page 112). What are the entries of the vector $A\mathbf{1}$? What are the entries of the vector $A^T\mathbf{1}$?

6.5 *Adjacency matrix of reversed graph.* Suppose A is the adjacency matrix of a directed graph (see page 112). The *reversed graph* is obtained by reversing the directions of all the edges of the original graph. What is the adjacency matrix of the reversed graph? (Express your answer in terms of A.)

6.6 *Matrix-vector multiplication.* For each of the following matrices, describe in words how x and $y = Ax$ are related. In each case x and y are n-vectors, with $n = 3k$.

(a) $A = \begin{bmatrix} 0 & 0 & I_k \\ 0 & I_k & 0 \\ I_k & 0 & 0 \end{bmatrix}$.

(b) $A = \begin{bmatrix} E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & E \end{bmatrix}$, where E is the $k \times k$ matrix with all entries $1/k$.

6.7 *Currency exchange matrix.* We consider a set of n currencies, labeled $1, \ldots, n$. (These might correspond to USD, RMB, EUR, and so on.) At a particular time the exchange or conversion rates among the n currencies are given by an $n \times n$ (exchange rate) matrix R, where R_{ij} is the amount of currency i that you can buy for one unit of currency j. (All entries of R are positive.) The exchange rates include commission charges, so we have $R_{ji}R_{ij} < 1$ for all $i \neq j$. You can assume that $R_{ii} = 1$.

Suppose $y = Rx$, where x is a vector (with nonnegative entries) that represents the amounts of the currencies that we hold. What is y_i? Your answer should be in English.

6.8 *Cash flow to bank account balance.* The T-vector c represents the cash flow for an interest bearing bank account over T time periods. Positive values of c indicate a deposit, and negative values indicate a withdrawal. The T-vector b denotes the bank account balance in the T periods. We have $b_1 = c_1$ (the initial deposit or withdrawal) and

$$b_t = (1 + r)b_{t-1} + c_t, \quad t = 2, \ldots, T,$$

where $r > 0$ is the (per-period) interest rate. (The first term is the previous balance plus the interest, and the second term is the deposit or withdrawal.)

Find the $T \times T$ matrix A for which $b = Ac$. That is, the matrix A maps a cash flow sequence into a bank account balance sequence. Your description must make clear what all entries of A are.

6.9 *Multiple channel marketing campaign.* Potential customers are divided into m market segments, which are groups of customers with similar demographics, *e.g.*, college educated women aged 25–29. A company markets its products by purchasing advertising in a set of n channels, *i.e.*, specific TV or radio shows, magazines, web sites, blogs, direct mail, and so on. The ability of each channel to deliver impressions or views by potential customers is characterized by the *reach matrix*, the $m \times n$ matrix R, where R_{ij} is the number of views of customers in segment i for each dollar spent on channel j. (We assume that the total number of views in each market segment is the sum of the views from each channel, and that the views from each channel scale linearly with spending.) The n-vector c will denote the company's purchases of advertising, in dollars, in the n channels. The m-vector v gives the total number of impressions in the m market segments due to the advertising in all channels. Finally, we introduce the m-vector a, where a_i gives the profit in dollars per impression in market segment i. The entries of R, c, v, and a are all nonnegative.

(a) Express the total amount of money the company spends on advertising using vector/matrix notation.

(b) Express v using vector/matrix notation, in terms of the other vectors and matrices.

(c) Express the total profit from all market segments using vector/matrix notation.

(d) How would you find the single channel most effective at reaching market segment 3, in terms of impressions per dollar spent?

(e) What does it mean if R_{35} is very small (compared to other entries of R)?

6.10 *Resource requirements.* We consider an application with n different job (types), each of which consumes m different resources. We define the $m \times n$ resource matrix R, with entry R_{ij} giving the amount of resource i that is needed to run one unit of job j, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. (These numbers are typically positive.) The number (or amount) of each of the different jobs to be processed or run is given by the entries of the n-vector x. (These entries are typically nonnegative integers, but they can be fractional if the jobs are divisible.) The entries of the m-vector p give the price per unit of each of the resources.

(a) Let y be the m-vector whose entries give the total of each of the m resources needed to process the jobs given by x. Express y in terms of R and x using matrix and vector notation.

(b) Let c be an n-vector whose entries gives the cost per unit for each job type. (This is the total cost of the resources required to run one unit of the job type.) Express c in terms of R and p using matrix and vector notation.

Remark. One example is a data center, which runs many instances of each of n types of application programs. The resources include number of cores, amount of memory, disk, and network bandwidth.

6.11 Let A and B be two $m \times n$ matrices. Under each of the assumptions below, determine whether $A = B$ must always hold, or whether $A = B$ holds only sometimes.

(a) Suppose $Ax = Bx$ holds for all n-vectors x.

(b) Suppose $Ax = Bx$ for some nonzero n-vector x.

6.12 *Skew-symmetric matrices.* An $n \times n$ matrix A is called *skew-symmetric* if $A^T = -A$, *i.e.*, its transpose is its negative. (A symmetric matrix satisfies $A^T = A$.)

(a) Find all 2×2 skew-symmetric matrices.

(b) Explain why the diagonal entries of a skew-symmetric matrix must be zero.

(c) Show that for a skew-symmetric matrix A, and any n-vector x, $(Ax) \perp x$. This means that Ax and x are orthogonal. *Hint.* First show that for any $n \times n$ matrix A and n-vector x, $x^T(Ax) = \sum_{i,j=1}^{n} A_{ij}x_ix_j$.

(d) Now suppose A is any matrix for which $(Ax) \perp x$ for any n-vector x. Show that A must be skew-symmetric. *Hint.* You might find the formula

$$(e_i + e_j)^T(A(e_i + e_j)) = A_{ii} + A_{jj} + A_{ij} + A_{ji},$$

valid for any $n \times n$ matrix A, useful. For $i = j$, this reduces to $e_i^T(Ae_i) = A_{ii}$.

6.13 *Polynomial differentiation.* Suppose p is a polynomial of degree $n - 1$ or less, given by $p(t) = c_1 + c_2t + \cdots + c_nt^{n-1}$. Its derivative (with respect to t) $p'(t)$ is a polynomial of degree $n-2$ or less, given by $p'(t) = d_1 + d_2t + \cdots + d_{n-1}t^{n-2}$. Find a matrix D for which $d = Dc$. (Give the entries of D, and be sure to specify its dimensions.)

6.14 *Norm of matrix-vector product.* Suppose A is an $m \times n$ matrix and x is an n-vector. A famous inequality relates $\|x\|$, $\|A\|$, and $\|Ax\|$:

$$\|Ax\| \leq \|A\|\|x\|.$$

The left-hand side is the (vector) norm of the matrix-vector product; the right-hand side is the (scalar) product of the matrix and vector norms. Show this inequality. *Hints.* Let a_i^T be the ith row of A. Use the Cauchy–Schwarz inequality to get $(a_i^T x)^2 \leq \|a_i\|^2\|x\|^2$. Then add the resulting m inequalities.

6.15 *Distance between adjacency matrices.* Let A and B be the $n \times n$ adjacency matrices of two directed graphs with n vertices (see page 112). The squared distance $\|A - B\|^2$ can be used to express how different the two graphs are. Show that $\|A - B\|^2$ is the total number of directed edges that are in one of the two graphs but not in the other.

6.16 *Columns of difference matrix.* Are the columns of the difference matrix D, defined in (6.5), linearly independent?

6.17 *Stacked matrix.* Let A be an $m \times n$ matrix, and consider the stacked matrix S defined by

$$S = \begin{bmatrix} A \\ I \end{bmatrix}.$$

When does S have linearly independent columns? When does S have linearly independent rows? Your answer can depend on m, n, or whether or not A has linearly independent columns or rows.

6.18 *Vandermonde matrices.* A Vandermonde matrix is an $m \times n$ matrix of the form

$$V = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^{n-1} \end{bmatrix}$$

where $t_1, \ldots, t_m$ are numbers. Multiplying an n-vector c by the Vandermonde matrix V is the same as evaluating the polynomial of degree less than n, with coefficients $c_1, \ldots, c_n$, at the points $t_1, \ldots, t_m$; see page 120. Show that the columns of a Vandermonde matrix are linearly independent if the numbers $t_1, \ldots, t_m$ are distinct, *i.e.*, different from each other. *Hint.* Use the following fact from algebra: If a polynomial p with degree less than n has n or more roots (points t for which $p(t) = 0$) then all its coefficients are zero.

6.19 *Back-test timing.* The $T \times n$ asset returns matrix R gives the returns of n assets over T periods. (See page 120.) When the n-vector w gives a set of portfolio weights, the T-vector Rw gives the time series of portfolio return over the T time periods. Evaluating portfolio return with past returns data is called *back-testing*.

Consider a specific case with $n = 5000$ assets, and $T = 2500$ returns. (This is 10 years of daily returns, since there are around 250 trading days in each year.) About how long would it take to carry out this back-test, on a 1 Gflop/s computer?

6.20 *Complexity of matrix-vector multiplication.* On page 123 we worked out the complexity of computing the m-vector Ax, where A is an $m \times n$ matrix and x is an n-vector, when each entry of Ax is computed as an inner product of a row of A and the vector x. Suppose instead that we compute Ax as a linear combination of the columns of A, with coefficients $x_1, \ldots, x_n$. How many flops does this method require? How does it compare to the method described on page 123?

6.21 *Complexity of matrix-sparse-vector multiplication.* On page 123 we consider the complexity of computing Ax, where A is a sparse $m \times n$ matrix and x is an n-vector x (not assumed to be sparse). Now consider the complexity of computing Ax when the $m \times n$ matrix A is not sparse, but the n-vector x is sparse, with $\mathbf{nnz}(x)$ nonzero entries. Give the total number of flops in terms of m, n, and $\mathbf{nnz}(x)$, and simplify it by dropping terms that are dominated by others when the dimensions are large. *Hint.* The vector Ax is a linear combination of $\mathbf{nnz}(x)$ columns of A.

6.22 *Distribute or not?* Suppose you need to compute $z = (A + B)(x + y)$, where A and B are $m \times n$ matrices and x and y are n-vectors.

(a) What is the approximate flop count if you evaluate z as expressed, *i.e.*, by adding A and B, adding x and y, and then carrying out the matrix-vector multiplication?

(b) What is the approximate flop count if you evaluate z as $z = Ax + Ay + Bx + By$, *i.e.*, with four matrix-vector multiplies and three vector additions?

(c) Which method requires fewer flops? Your answer can depend on m and n. *Remark.* When comparing two computation methods, we usually do not consider a factor of 2 or 3 in flop counts to be significant, but in this exercise you can.