Lecture Notes
to be used in conjunction with

233 Computational Techniques

István Maros

Department of Computing
Imperial College London

V2.9e
January 2008

Contents

1 Introduction

To solve a wide range of important problems we need advanced computational techniques. In general, these techniques are domain specific but many of them share some basic techniques and methods. They are all suitable for computer implementation and, indeed, the referred problems require considerable computing power to solve them. The lectures (and the notes) present some of the techniques with the intent to further develop your skills and, at the same time, prepare you for your 3rd and 4th year studies.

2 Computation and Accuracy

2.1 Measuring Errors

Most computational algorithms involve calculations in which precision is limited and, therefore, the results may not be exact. It is important to develop ideas and measurements to characterize the difference between an exact and a computed result.

If a quantity $\bar{x}$ is viewed as an approximation to x, the *absolute error* e_A is defined as:

$$e_A = |x - \bar{x}|.$$

Example 1 *Let* $x = 0.0314$ *and* $\bar{x} = 0.0315$*. The absolute error is*

$$e_A = |0.0314 - 0.0315| = 0.0001 = 10^{-4}.$$

Example 2 *As another example, let* $x = 10^{10}$ *and* $\bar{x} = 10^{10} + 100$*. Now, the absolute error is*

$$e_A = |10^{10} - (10^{10} + 100)| = 100 = 10^2.$$

Though the absolute error of example 1 is much smaller than that of example 2, the second approximation is more accurate because it agrees with the number in eight significant digits.

A calibration based on the size of x can be achieved by defining the *relative error* if $x \neq 0$:

$$e_R = \frac{e_A}{|x|} = \frac{|x - \bar{x}|}{|x|}.$$

Example 3 *Applying the definition of* e_R*, the relative error of example 1 is*

$$e_R = \frac{|0.0314 - 0.0315|}{|0.0314|} = \frac{0.0001}{0.0314} \approx 3.18 \times 10^{-2},$$

while the relative error of example 2 is

$$e_R = \frac{|10^{10} - (10^{10} + 100)|}{|10^{10}|} = 10^{-8},$$

which confirms what was expected, namely, the second approximation is better than the first.

If the number x is zero then the relative error is not defined. Additionally, if x is small the relative error can be misleading and the absolute error should be used instead. To avoid the need to choose between absolute and relative error, a *mixed error measure* is often used in practice:

$$e = \frac{e_A}{1 + |x|} = \frac{|x - \bar{x}|}{1 + |x|}.$$

The value of e is similar to the relative error when $|x| \gg 1$ and to the absolute error when $|x| \ll 1$.

2.2 Floating-Point Representation of Numbers

In nearly all scientific, engineering and business applications that require computing, calculations with real numbers are performed using *floating-point arithmetic* of a computer. Real numbers in a computer are represented by a finite number of bits which immediately entails that not every number can be represented exactly. Additionally, the result of an arithmetic operation with two representable numbers may result in a non-representable number.

Given an integer $b \geq 2$, any real number r can be written in base b as

$$r = \pm m \times b^z, \tag{1}$$

where z (the *exponent*) is a signed integer and m (the *mantissa*) is a nonnegative real number. For a general r, m may contain an infinite number of digits, e.g., if $b = 10$ and $r = 1/3$. Most computers are designed with $b = 2$ or $b = 16$ (*binary* or *hexadecimal* arithmetic), while humans use $b = 10$ (*decimal* arithmetic).

It is easy to see that the above representation is not unique. Any given number r can be represented in several different ways even if the base is the same. For example, 1.23 can be written as 1.23×10^0, or 0.123×10^1, or 123.0×10^{-2}, etc.

In order to make m unique when $r \neq 0$, we require that $b^{-1} \leq m < 1$ in the representation (1). Such a number representation is called *normalized*. When r is represented on a computer, only finite number of digits of m are retained:

$$r = \pm .m_1 m_2 \ldots m_t \times b^z, \tag{2}$$

where t is the number of digits in the mantissa and each m_i is a valid digit in the base b number system: $0 \leq m_i < b$. In a normalized nonzero floating-point number (fl-number), $m_1 \neq 0$. In general, t is fixed for a particular machine and level of precision.

Example 4 *Valid and invalid normalized numbers:*

Type	*Valid*	*Invalid*
binary ($t = 15$)	$-.110011101101001 \times 2^3$	$-.110011121101001 \times 2^3$
decimal ($t = 4$)	$.1234 \times 10^{-2}$	$.0123 \times 10^{-1}$
hexadecimal ($t = 4$)	$-.AB7F \times 16^5$	$-.0FG7 \times 16^5$

On most scientific computers fl-numbers may be represented in single and/or double precision. The IEEE standard for floating-point arithmetic uses $b = 2$ in (2). The value of t is 23 for single precision (SP) and 52 for double precision (DP). SP and DP numbers occupy 32 and 64 bits, respectively. Since one bit is needed for the sign of the mantissa, 8 and 11 bits are left for the exponents. The smallest and largest representable absolute values are:

Precision	Smallest absolute	Largest absolute
Single	$\approx 3.40282 \times 10^{-38}$	$\approx 3.40282 \times 10^{38}$
Double	$\approx 1.79769 \times 10^{-308}$	$\approx 1.79769 \times 10^{308}$

These figures are by a factor of 2 larger than obtainable by (2). The 'trick' is that IEEE assumes there is a fixed bit of 1 before the binary point and such $r = \pm 1.m_1 m_2 \ldots m_t \times 2^z$. If the exponent of a number is smaller then the smallest representable exponent (in decimal: -38 for SP, -308 for DP) then an *underflow* occurs and the number is usually set to zero. If the exponent of a number is greater then the largest representable exponent (in decimal: $+38$ for SP, $+308$ for DP) then an *overflow* occurs which is usually a fatal error resulting in the termination of the computations.

The t-digit mantissa allows for ≈ 7 significant decimal digits in SP, and ≈ 16 in DP.

When we perform computations with any number x, it must first be converted to its machine representable floating-point equivalent $\bar{x} = fl(x)$. If x itself is representable then $x = fl(x)$. Assuming that overflow or underflow do not occur, $\bar{x}$ is usually obtained by either *rounding* or *truncating (chopping)* the representation (1) of x. The *relative accuracy* a_R of $fl(x)$ is b^{1-t} for truncated arithmetic and $\frac{1}{2}b^{1-t}$ for rounded arithmetic.

2.3 Arithmetic Operations

Errors occur not only in representing numbers on a computer but also in performing arithmetic operations. Even in the simplest possible case, the exact result with two representable numbers may not be representable. For instance, numbers 2.5 and 0.003141 are both representable on a hypothetical four digit decimal machine, their exact sum, 2.503141 is not and would be rounded or truncated to 2.503.

On machines with reasonable numerical properties, the stored result of a floating-point operation on two representable numbers x and y satisfies

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1 + \delta),$$

where 'op' is one of the basic operations '+', '−', '×', or '/' and $|\delta| \leq a_R$.

The work associated with the computation of the form $fl(u \times v + w)$ is called a *flop*. This operation includes access to the values of u, v and w, a multiplication, an addition and storage of the computed result.

Some laws of mathematics cannot be applied to floating-point calculations. This is due to the errors in representation and arithmetic. In particular, the associative property of exact addition is not satisfied for fl-addition. For instance, there exist representable numbers u, v and w such that

$$fl(fl(u + v) + w) \neq fl(u + fl(v + w)),$$

in other words, the result of fl-addition of three numbers may depend on the order they are added.

Example 5 *Let* $u = -1,\ v = 1,\ w = 0.0001$ *in a four-digit decimal arithmetic. Now*

$$fl(u + v) = 0.0, \quad fl(fl(u + v) + w) = 0.0001,$$

but

$$fl(v + w) = 1.0, \quad fl(u + fl(v + w)) = 0.0.$$

Similarly, mathematical identities such as $\left(\sqrt{|x|}\right)^2 = |x|$ may not hold for the floating-point representation of $\sqrt{x}$ and x.

An interesting example is the case of the famous harmonic series

$$1 + \frac{1}{2} + \frac{1}{3} + \ldots,$$

which is known to diverge with exact arithmetic. However, in finite arithmetic, the next term to be added eventually becomes so small that it does not alter the computed partial sum, so the series 'converges'.

3 Computational Linear Algebra

3.1 Vectors and Matrices

3.1.1 Vectors

In the present context, a *vector* is a linearly ordered set of real numbers and is denoted by a boldface lower case letter, like $\mathbf{v}$. The numbers are the *components* of the vector. Traditionally, vectors are represented in columnar form:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

The number of components determines the *dimension* of the vector. We say that $\mathbf{v}$ is an m dimensional vector, or an element of the m dimensional space of real numbers: $\mathbf{v} \in \mathbb{R}^m$.

If a column vector $\mathbf{v}$ is written rowwise, it is called the *transpose* of $\mathbf{v}$ and is denoted by superscript ^{T}, like $\mathbf{v}^T$. Therefore, $\mathbf{v}^T$ is a *row vector*: $\mathbf{v}^T = [v_1, v_2, \ldots, v_m]$. In some cases we use a different notation for row vectors.

If all the components of a vector are zero then it is called a *zero vector.* In any dimension there is a zero vector.

In $\mathbb{R}^m$ there are m *unit vectors.* The ith unit vector (usually denoted by $\mathbf{e}_i$) has a 1 in position i and zero elsewhere.

Example 6 *In $\mathbb{R}^3$ the zero vector and the three unit vectors are as follows:*

$$\mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

A vector $\mathbf{v}$ can be multiplied by a scalar, say λ. It is defined as

$$\lambda\mathbf{v} = \begin{bmatrix} \lambda v_1 \\ \lambda v_2 \\ \vdots \\ \lambda v_m \end{bmatrix},$$

that is, every component of $\mathbf{v}$ is multiplied by λ.

Addition of two vectors is defined if both are of the same dimension. The sum of $\mathbf{u}$ and $\mathbf{v}$ is:

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_m + v_m \end{bmatrix},$$

that is, components with the same index are added.

Example 7 *Compute $\mathbf{u} + 2.5\mathbf{v}$ with the following vectors:*

$$\mathbf{u} = \begin{bmatrix} 11 \\ -5 \\ 2 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 11 \\ -5 \\ 2 \end{bmatrix} + 2.5 \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \\ 2 \end{bmatrix}.$$

Ordering of vectors. Two vectors $\mathbf{u}$ and $\mathbf{v}$ are *equal* if they are of the same dimension, say m, and all their corresponding components are equal, that is, $u_i = v_i, \ i = 1, \ldots, m$.

We write $\mathbf{u} \geq \mathbf{0}$ if all components of $\mathbf{u}$ are nonnegative: $u_i \geq 0, \ \forall i$. Two vectors, say $\mathbf{u}$ and $\mathbf{v}$, of the same dimension can be compared and we say that $\mathbf{u} \geq \mathbf{v}$ if $\mathbf{u} - \mathbf{v} \geq \mathbf{0}$. If this relation does not hold no relationship exists between them under this definition.

Lexicographic ordering of vectors. This is a generalization of the traditional ordering. A nonzero vector $\mathbf{u}$ is said to be *lexicographically positive* (l-positive) if its first nonzero component is positive. Notationally it is expressed as $\mathbf{u} \succ \mathbf{0}$. The interpretation of lexicographical negativity is obvious. For example, $\mathbf{u}_1 = [0, 0, 2, -1]^T$ is l-positive, $\mathbf{u}_1 \succ \mathbf{0}$, while $\mathbf{u}_2 = [0, -1, 0, 3]^T$ is l-negative, $\mathbf{u}_2 \prec \mathbf{0}$.

Vector $\mathbf{u}$ is *lexicographically greater* than $\mathbf{v}$ if they are of the same dimension, $\mathbf{u} \neq \mathbf{v}$ and $\mathbf{u} - \mathbf{v} \succ \mathbf{0}$. It is also written as $\mathbf{u} \succ \mathbf{v}$. If $\mathbf{u} \neq \mathbf{v}$ than one of them is lexicographically greater than the other.

$\mathbf{u} \succeq \mathbf{v}$ means that either $\mathbf{u} = \mathbf{v}$ or $\mathbf{u} \succ \mathbf{v}$ holds. It is easy to see that $\mathbf{u} \succeq \mathbf{v}$ and $\mathbf{u} \preceq \mathbf{v}$ imply $\mathbf{u} = \mathbf{v}$.

Linear combination of vectors. Given a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of the same dimension and a set of scalars $\lambda_1, \lambda_2, \ldots, \lambda_k$, the *linear combination, LC*, of the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ is defined as:

$$\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \cdots + \lambda_k \mathbf{v}_k = \sum_{j=1}^{k} \lambda_j \mathbf{v}_j. \tag{3}$$

If all scalars $\lambda_j = 0$ then (3) is the *trivial* linear combination. If at least one λ_j is different from zero then (3) is a *non-trivial* linear combination.

The *line segment* connecting two vectors $\mathbf{x}$ and $\mathbf{y}$ of the same dimension consists of all vectors of the form $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ with $0 \leq \lambda \leq 1$. This is called the *convex linear combination* of $\mathbf{x}$ and $\mathbf{y}$. The convex linear combination of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is defined as

$$\sum_{i=1}^{n} \lambda_i \mathbf{x}_i = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_n \mathbf{x}_n, \quad \text{with } \lambda_i \geq 0 \text{ for } i = 1, \ldots, n \text{ and } \sum_{i=1}^{n} \lambda_i = 1.$$

The multiplication of two vectors $\mathbf{u}$ and $\mathbf{v}$ of the same dimension is called *inner, dot*, or *scalar product.* It is denoted by $\mathbf{u}^T\mathbf{v}$ (sometimes by $\langle \mathbf{u}, \mathbf{v} \rangle$) and is defined in the following way:

$$\mathbf{u}^T\mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_m v_m = \sum_{i=1}^{m} u_i v_i,$$

which is the sum of the pairwise products of the components. Since $\mathbf{u}^T$ is a row vector, the inner product can be conceived as moving rowwise in the first component and columnwise in the second.

The dot product evidently satisfies *commutativity*, $\mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u}$, and *distributivity over vector addition*, $\mathbf{u}^T(\mathbf{v} + \mathbf{w}) = \mathbf{u}^T\mathbf{v} + \mathbf{u}^T\mathbf{w}$.

It is worth remembering that the result of the dot product is a scalar (number). This number can be zero even if none the vectors is zero, as shown in the following example.

Example 8 *Dot product of two nonzero vectors is equal to zero:*

$$\textit{Let } \mathbf{u} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}, \qquad \textit{then} \quad \mathbf{u}^T\mathbf{v} = 4 \times 1 + 1 \times 2 + 3 \times (-2) = 0.$$

If the $\mathbf{u}^T\mathbf{v}$ dot product of two nonzero vectors is zero, then $\mathbf{u}$ and $\mathbf{v}$ are said to be *orthogonal*, $\mathbf{u} \perp \mathbf{v}$.

3.1.2 Matrices

The notion of a matrix was introduced by English mathematician Arthur Cayley (1821–1895) in 1857.

A matrix is a rectangular array of numbers. It is characterized by the number of rows and columns. A matrix is denoted by a boldface capital letter, its elements by the corresponding lower case letter. The rectangular structure is enclosed in square brackets. If $\mathbf{A}$ has m rows and n columns it looks as follows:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

We say that matrix $\mathbf{A}$ is an element of the $m \times n$ dimensional space of real numbers: $\mathbf{A} \in \mathbb{R}^{m \times n}$. The row dimension of $\mathbf{A}$ is m, the column dimension is n. A matrix element a_{ij} is identified by a lower case letter and a double subscript. The subscript expresses that a_{ij} is at the intersection of row i and column j.

Example 9 *A matrix with 2 rows and 3 columns:*

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & -1 \\ 9 & 0 & 2 \end{bmatrix}, \qquad \mathbf{A} \in \mathbb{R}^{2 \times 3}$$

If $m = n$, the matrix is called *square*. Two matrices $\mathbf{A}$ and $\mathbf{B}$ have the same dimension if the row dimension of $\mathbf{A}$ is equal to the row dimension of $\mathbf{B}$ and their column dimensions are also equal. If all the elements of a matrix are equal to 0 then the matrix is a *zero matrix*. A zero matrix can be of any size.

Example 10 *This is the* 2×4 *zero matrix:*

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Elements of the matrix with equal row and column index are called *diagonal elements*, like a_{22}, or a_{ii}. Note that the definition of the diagonal elements does not require that the matrix is square.

Example 11 *If* $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$ *then the diagonal elements are* $a_{11} = 1$ *and* $a_{22} = 6$.

Two matrices $\mathbf{A}$ and $\mathbf{B}$ are said to be equal if they have the same dimension and $a_{ij} = b_{ij}$ for all elements.

If $\mathbf{A}$ is an $m \times n$ matrix its *transpose* is an $n \times m$ matrix, denoted by $\mathbf{A}^T$, obtained from $\mathbf{A}$ by interchanging its rows and columns:

$$a_{ij}^T = a_{ji}, \quad 1 \leq i \leq n, \ 1 \leq j \leq m,$$

which means that the first row of $\mathbf{A}^T$ is equal to the first column of $\mathbf{A}$, and so on.

Example 12

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & -1 \\ 9 & 0 & 2 \end{bmatrix}, \quad \mathbf{A}^T = \begin{bmatrix} 1 & 9 \\ 3 & 0 \\ -1 & 2 \end{bmatrix}.$$

The transpose of the transpose is the original matrix:

$$(\mathbf{A}^T)^T = \mathbf{A}.$$

The diagonal elements of $\mathbf{A}$ and $\mathbf{A}^T$ are the same. The matrix $\mathbf{A}$ is said to be *symmetric* if $\mathbf{A} = \mathbf{A}^T$. It entails that a symmetric matrix is square and $a_{ij} = a_{ji}$ for all i and j.

It is often convenient to partition a matrix into submatrices. This is indicated by drawing partitioning lines through the matrix, as for example,

$$\mathbf{A} = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ \hline a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{array}\right] = \left[\begin{array}{cc} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right]$$

The resulting submatrices are usually denoted by $\mathbf{A}_{ij}$, as illustrated.

A matrix $\mathbf{A}$ can be conceived as a set of column vectors. Column j of $\mathbf{A}$ is denoted by $\mathbf{a}_j$ and $\mathbf{A}$ can be written as

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n] . \tag{4}$$

We can say that $\mathbf{A}$ consists of n m-dimensional vectors.

Similarly, $\mathbf{A}$ can be considered as a set of row vectors. Row i of $\mathbf{A}$ is denoted by $\mathbf{a}^i$, that is, the index is in the superscript to indicate that this is a row vector. In this way

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}^1 \\ \mathbf{a}^2 \\ \vdots \\ \mathbf{a}^m \end{bmatrix} \tag{5}$$

and we say that $\mathbf{A}$ consists of m n-dimensional row vectors.

We can perform several *operations on matrices*. The simplest operation is the computation of a scalar multiple of a matrix. Let λ be a scalar and $\mathbf{A} \in \mathbb{R}^{m \times n}$, then

$$\lambda\mathbf{A} = \begin{bmatrix} \lambda a_{11} & \lambda a_{12} & \ldots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \ldots & \lambda a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{m1} & \lambda a_{m2} & \ldots & \lambda a_{mn} \end{bmatrix},$$

that is, every element of $\mathbf{A}$ is multiplied by λ.

Example 13 *If* $\lambda = 2$ *and* $\mathbf{A} = \begin{bmatrix} 1 & 3 & -1 \\ 9 & 0 & 2 \end{bmatrix}$, *then* $\lambda\mathbf{A} = 2\mathbf{A} = \begin{bmatrix} 2 & 6 & -2 \\ 18 & 0 & 4 \end{bmatrix}$.

The sum of two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined if both have the same dimension (say $m \times n$). In this case:

$$\mathbf{C} = \mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \ldots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \ldots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \ldots & a_{mn} + b_{mn} \end{bmatrix},$$

that is, $c_{ij} = a_{ij} + b_{ij}$ for all i and j. An immediate consequence is that $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$ (commutativity).

Example 14

$$\begin{bmatrix} 1 & 2 & 3 \\ -1 & -2 & 0 \end{bmatrix} + \begin{bmatrix} -1 & 3 & -1 \\ 0 & 2 & 10 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 2 \\ -1 & 0 & 10 \end{bmatrix}$$

3.1.3 More about vectors and matrices

The *product of matrix* $\mathbf{A}$ *and vector* $\mathbf{x}$, $\mathbf{Ax}$, is defined if $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{x} \in \mathbb{R}^n$, that is, the column dimension of $\mathbf{A}$ is equal to the dimension of $\mathbf{x}$. The result is an m dimensional vector $\mathbf{y} = \mathbf{Ax}$. Because of this, we can say that by the $\mathbf{Ax}$ product the $m \times n$ dimensional matrix $\mathbf{A}$ transforms the n dimensional vector $\mathbf{x}$ into an m dimensional vector $\mathbf{y}$. The ith component of $\mathbf{y}$ is as:

$$y_i = \sum_{j=1}^{n} a_{ij} x_j$$

This is nothing but the dot product of row i of $\mathbf{A}$ with $\mathbf{x}$, that is, $y_i = \mathbf{a}^i\mathbf{x}$. In this way:

$$\mathbf{y} = \begin{bmatrix} \mathbf{a}^1\mathbf{x} \\ \mathbf{a}^2\mathbf{x} \\ \vdots \\ \mathbf{a}^m\mathbf{x} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} a_{1j}x_j \\ \sum_{j=1}^{n} a_{2j}x_j \\ \vdots \\ \sum_{j=1}^{n} a_{mj}x_j \end{bmatrix}. \tag{6}$$

The $\mathbf{Ax}$ product can also be considered as the linear combination of the columns of $\mathbf{A}$ with the set of scalars $x_1, x_2, \ldots, x_n$:

$$\mathbf{y} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n = \sum_{j=1}^{n} x_j\mathbf{a}_j. \tag{7}$$

Of course, the two forms, (6) and (7), are algebraically equivalent.

Example 15 *Matrix representation of problems.*

A company produces three types of alloys: soft (S), medium (M) and hard (H). They are produced as blends of two basic components C and K. To produce one ton of S, 0.75 t of C and 0.25 t of K are needed. The requirements for M and H are 0.55, 0.45 and 0.3, 0.7, respectively. The data can be represented in matrix form:

	Alloy		
Component	*S*	*M*	*H*
C	*0.75*	*0.55*	*0.30*
K	*0.25*	*0.45*	*0.70*

from which $\mathbf{A} = \begin{bmatrix} 0.75 & 0.55 & 0.30 \\ 0.25 & 0.45 & 0.70 \end{bmatrix}$

The company wants to know what amounts of the components are needed to fulfil an order for 10 tons of S, 12 tons of M, and 20 tons of H. If the order is formed as vector $\mathbf{x} = [10, 12, 20]^T$ *then the answer to the question is* $\mathbf{y} = \mathbf{Ax}$ *which is*

$$\begin{bmatrix} 0.75 & 0.55 & 0.30 \\ 0.25 & 0.45 & 0.70 \end{bmatrix} \begin{bmatrix} 10 \\ 12 \\ 20 \end{bmatrix} = \begin{bmatrix} 10 \times 0.75 + 12 \times 0.55 + 20 \times 0.30 \\ 10 \times 0.25 + 12 \times 0.45 + 20 \times 0.70 \end{bmatrix} = \begin{bmatrix} 20.1 \\ 21.9 \end{bmatrix}.$$

Therefore, 20.1 tons of C and 21.9 tons of K are needed.

The product of two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined for the case when the column dimension of $\mathbf{A}$ is equal to the row dimension of $\mathbf{B}$ (the two matrices are *conformable* for multiplication). If $\mathbf{A} \in \mathbb{R}^{m\times p}$ and $\mathbf{B} \in \mathbb{R}^{p\times n}$ the resulting $\mathbf{C} = \mathbf{AB}$ will be an $m \times n$ matrix with c_{ij} defined by:

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}. \tag{8}$$

It is easy to notice that (8) is nothing but the dot product of row i of $\mathbf{A}$ and column j of $\mathbf{B}$: $c_{ij} = \mathbf{a}^i\mathbf{b}_j$ which is well defined since both are p dimensional vectors.

In the case of $\mathbf{AB}$ we say that $\mathbf{A}$ *premultiplies* $\mathbf{B}$ or $\mathbf{B}$ *postmultiplies* $\mathbf{A}$.

Example 16 *Matrix-matrix products:*

$$\textit{If}\quad \mathbf{A}=\begin{bmatrix}1 & 2 & 3\\ 1 & -1 & 1\end{bmatrix},\quad \mathbf{B}=\begin{bmatrix}-1 & 2\\ 2 & -1\\ -3 & 0\end{bmatrix},\quad \mathbf{C}=\begin{bmatrix}2 & -1\\ 1 & 1\end{bmatrix}$$

$$\textit{then}\quad \mathbf{AB}=\begin{bmatrix}-6 & 0\\ -6 & 3\end{bmatrix},\quad \mathbf{BC}=\begin{bmatrix}0 & 3\\ 3 & -3\\ -6 & 3\end{bmatrix},\quad \mathbf{CA}=\begin{bmatrix}1 & 5 & 5\\ 2 & 1 & 4\end{bmatrix}$$

The transpose of a product of two matrices can be expressed as the product of their transposes taken in reverse order:

$$(\mathbf{AB})^T=\mathbf{B}^T\mathbf{A}^T.$$

The transpose of a product of a matrix and a vector, assuming conformability,

$$(\mathbf{Ax})^T=\mathbf{x}^T\mathbf{A}^T,\quad\text{and}\quad(\mathbf{A}^T\mathbf{y})^T=\mathbf{y}^T\mathbf{A}.$$

Some matrices with special structure play important role in linear algebra. The *unit matrix* is a square matrix with all off-diagonal elements being zero and all diagonal elements 1. In case of square matrices it is customary to quote just the column dimension (which is equal to the row dimension). A unit matrix is denoted by $\mathbf{I}$ and sometimes its dimension is given in subscript, like $\mathbf{I}_m$. An important feature of the unit matrix is that if it pre- or post-multiplies a conformable matrix $\mathbf{A}$ then the resulting matrix is just $\mathbf{A}$. The m dimensional unit matrix can be considered as the collection of the m dimensional unit vectors, or m dimensional unit row vectors.

$$\mathbf{I}_m=\begin{bmatrix}1 & 0 & \dots & 0\\ 0 & 1 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & 1\end{bmatrix}$$

The existence of $\mathbf{AB}$ does not necessarily mean that $\mathbf{BA}$ also exists because of the conformability requirement. Even if both $\mathbf{AB}$ and $\mathbf{BA}$ exist (both are square and of the same dimension), it is generally true that $\mathbf{AB}\neq\mathbf{BA}$. Equality holds in special cases only, like $\mathbf{AI}_m=\mathbf{I}_m\mathbf{A}$ or, if $\mathbf{0}$ denotes the $m\times m$ dimensional zero matrix then, $\mathbf{A0}=\mathbf{0A}$.

If $\mathbf{A},\ \mathbf{B},\ \mathbf{C}$ and $\mathbf{D}$ have appropriate dimensions then the following relations hold:

$$\begin{aligned}\mathbf{A}(\mathbf{BC})&=(\mathbf{AB})\mathbf{C} &&\text{(Associativity)}\\ \mathbf{A}(\mathbf{B}+\mathbf{D})&=\mathbf{AB}+\mathbf{AD} &&\text{(Distributivity)}\\ (\mathbf{B}+\mathbf{D})\mathbf{C}&=\mathbf{BC}+\mathbf{DC} &&\text{(Distributivity)}\end{aligned}$$

The *power* of a square matrix $\mathbf{A}$ is defined as repeated multiplication:

$$\mathbf{A}^2=\mathbf{AA},\ \mathbf{A}^3=\mathbf{AA}^2,\ \dots,\ \mathbf{A}^n=\mathbf{AA}^{n-1}=\underbrace{\mathbf{A}\cdots\mathbf{A}}_{n\text{ times}}.$$

If $n=0$ then $\mathbf{A}^n$ is defined to be $\mathbf{I}$: $\mathbf{A}^0=\mathbf{I}$:

A matrix is *diagonal* if all its off-diagonal elements are zero. If $\mathbf{D}$ is a square diagonal matrix, it can be defined by the list of the diagonal elements, $\mathbf{D}=\langle d_1,d_2,\dots,d_m\rangle$, or $\operatorname{diag}(d_1,d_2,\dots,d_m)$, or simply $\operatorname{diag}(d_i)$. Note that the diagonal elements are not necessarily nonzero.

Example 17

$$\mathbf{D}=\begin{bmatrix}-2 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 3\end{bmatrix}=\langle-2,0,3\rangle=\operatorname{diag}(-2,0,3)$$

Premultiplication by a diagonal matrix scales the rows, postmultiplication by a diagonal matrix scales the columns, since if $\mathbf{D}$ and $\bar{\mathbf{D}}$ are diagonal of appropriate dimension, $\mathbf{A}$ is $m \times n$ and using the row and column vector representations of $\mathbf{A}$, (5) and (4), respectively, then

$$\mathbf{DA} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_m \end{bmatrix} \begin{bmatrix} \mathbf{a}^1 \\ \mathbf{a}^2 \\ \vdots \\ \mathbf{a}^m \end{bmatrix} = \begin{bmatrix} d_1\mathbf{a}^1 \\ d_2\mathbf{a}^2 \\ \vdots \\ d_m\mathbf{a}^m \end{bmatrix},$$

and

$$\mathbf{A}\bar{\mathbf{D}} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \begin{bmatrix} \bar{d}_1 & 0 & \dots & 0 \\ 0 & \bar{d}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{d}_n \end{bmatrix} = [\bar{d}_1\mathbf{a}_1, \bar{d}_2\mathbf{a}_2, \dots, \bar{d}_n\mathbf{a}_n],$$

Next, we discuss *triangular* matrices as they are of great importance in computational linear algebra. A square matrix $\mathbf{U}$ is said to be *upper triangular* if all elements below the diagonal are zero:

$$u_{ij} = 0 \text{ for all } i > j.$$

In a similar fashion, a square matrix $\mathbf{L}$ is *lower triangular* if all its elements above the diagonal are zero:

$$\ell_{ij} = 0 \text{ for all } i < j.$$

Either form of triangular matrix is called *unit triangular* if all its diagonal elements are 1.

Example 18 $\mathbf{U}$ *is upper triangular and* $\mathbf{L}$ *is lower unit triangular:*

$$\mathbf{U} = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 4 & 1 \\ 0 & 0 & 9 \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & 6 & 1 \end{bmatrix}.$$

Recall, two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthogonal if $\mathbf{u}^T\mathbf{v} = 0$. A set of nonzero m-vectors

$$\mathbf{v}_1, \dots, \mathbf{v}_n \tag{9}$$

is said to be orthogonal if

$$\mathbf{v}_i^T\mathbf{v}_j = 0 \text{ for all } i \neq j.$$

Furthermore, if each of these vectors has the property that

$$\mathbf{v}_i^T\mathbf{v}_i = 1 \text{ for } 1 \leq i \leq n \tag{10}$$

then the vectors are said to form and *orthonormal* set.

Example 19 $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *are orthonormal:*

$$\mathbf{v}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

The set of vectors in (9) can be considered a matrix $\mathbf{V}$. Assume, its column vectors are normalized as in (10). It is easy to see that with these vectors $\mathbf{V}$ satisfies

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}_n. \tag{11}$$

A square matrix with orthonormal columns is called an *orthogonal matrix.* An orthogonal matrix is usually denoted by $\mathbf{Q}$. In this case, a direct consequence of (11) is that

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

Assume, $\mathbf{a}$ and $\mathbf{b}$ are two vectors of arbitrary dimension. The *outer product* of them is denoted by $\mathbf{ab}^T$ and is interpreted in the following way:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}, \quad \mathbf{b}^T = [b_1, b_2, \dots, b_n], \quad \mathbf{ab}^T = \begin{bmatrix} a_1b_1 & a_1b_2 & \dots & a_1b_n \\ a_2b_1 & a_2b_2 & \dots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_mb_1 & a_mb_2 & \dots & a_mb_n \end{bmatrix}. \tag{12}$$

Row i in the matrix of the outer product (12) is nothing but a_i times $\mathbf{b}^T$ and column j is b_j times $\mathbf{a}$. If the dimensions of $\mathbf{a}$ and $\mathbf{b}$ are equal then $\mathbf{ab}^T$ is square.

If $\mathbf{u}$ and $\mathbf{v}$ are m-vectors, the *rank-one update* of the unit matrix is defined as $\mathbf{E} = \mathbf{I} - \mathbf{uv}^T$. For practical purposes, it is more useful to explicitly include a multiplier α which otherwise can be absorbed in any of the vectors. In the sequel we will use the following form:

$$\mathbf{E} = \mathbf{I} - \alpha\mathbf{uv}^T$$

This relation will be of fundamental importance in the Gaussian elimination (to come later). If $\mathbf{x}$ is an m-vector, the $\mathbf{Ex}$ product can be computed without needing the matrix in explicit form:

$$\mathbf{Ex} = (\mathbf{I} - \alpha\mathbf{uv}^T)\mathbf{x} = \mathbf{x} - \alpha\mathbf{u}(\mathbf{v}^T\mathbf{x}) = \mathbf{x} - \gamma\mathbf{u},$$

where $\gamma = \alpha(\mathbf{v}^T\mathbf{x})$. Therefore, multiplying $\mathbf{x}$ by $\mathbf{E}$ means subtracting a multiple of $\mathbf{u}$ from $\mathbf{x}$.

3.2 Norms

Norms are used to measure the size of a vector or matrix. A norm can tell us which vector or matrix is 'smaller' or 'larger'. Several different norms can be defined as will be demonstrated in the sequel.

3.2.1 Vector norms

A *vector norm* is denoted by $\|\cdot\|$. It satisfies the following three properties:

(i) for any nonzero $\mathbf{x}$, $\|\mathbf{x}\| > 0$,

(ii) for any scalar λ, $\|\lambda\mathbf{x}\| = |\lambda|\,\|\mathbf{x}\|$,

(iii) for any two vectors $\mathbf{x}$ and $\mathbf{y}$, the *triangle inequality* holds: $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$.

The three most commonly used vector norms are derived from the p-norm which is defined for an m-vector $\mathbf{x}$ as follows:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p\right)^{1/p} \tag{13}$$

For different values of p the following norms are obtained:

$p = 1:$	$\lVert \mathbf{x} \rVert_1 = \sum_{i=1}^{m} \lvert x_i \rvert$	ℓ_1 norm
$p = 2:$	$\lVert \mathbf{x} \rVert_2 = \sqrt{\sum_{i=1}^{m} x_i^2}$	ℓ_2 norm, Euclidean norm/length
$p = \infty:$	$\lVert \mathbf{x} \rVert_\infty = \max_{1 \le i \le m} \lvert x_i \rvert$	ℓ_∞ norm, infinity norm.

Example 20 *Study the following two examples.*

$$\mathbf{u} = \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix}, \quad \lVert \mathbf{u} \rVert_1 = 11, \quad \lVert \mathbf{u} \rVert_2 = 7, \qquad \lVert \mathbf{u} \rVert_\infty = 6$$

$$\mathbf{v} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \quad \lVert \mathbf{v} \rVert_1 = 12, \quad \lVert \mathbf{v} \rVert_2 = 7.07, \quad \lVert \mathbf{v} \rVert_\infty = 5$$

Observe that $\mathbf{v}$ *is larger than* $\mathbf{u}$ *by the* ℓ_1 *norm, smaller by the* ℓ_∞ *norm and the two are nearly equal by the Euclidean* (ℓ_2) *norm.*

It can be shown that for any vector $\mathbf{v}$ the following relationships hold: $\lVert \mathbf{v} \rVert_1 \ge \lVert \mathbf{v} \rVert_2 \ge \lVert \mathbf{v} \rVert_\infty$.

Computing the ℓ_2 norm may be a source of numerical troubles. In several (mostly statistical) applications the x_i values can be very large numbers and there can be many of them (m is also large). In single precision computation it may happen that while the magnitude of the norm is within the representable range, some x_i^2 or the sum under the square root can be out of range causing floating point overflow which is a fatal error. The problem can be overcome at the expense of some extra operations. Since for any $\lambda > 0$, $\lVert \mathbf{x} \rVert_2 = \lambda \lVert (1/\lambda)\mathbf{x} \rVert_2$, we can write

$$\lVert \mathbf{x} \rVert_2 = \lambda \left(\sum_{i=1}^{m} \left(\frac{x_i}{\lambda} \right)^2 \right)^{1/2}$$

If $\lambda = \lVert \mathbf{x} \rVert_\infty = \max_i \{\lvert x_i \rvert\}$ is chosen then all $\left(\frac{x_i}{\lambda} \right)^2 \le 1$ and the norm becomes computable.

For any norm $\lVert \cdot \rVert$ and any nonzero vector $\mathbf{v}$ there exists a *normalized* vector $\mathbf{u}$ which has the same direction as $\mathbf{v}$ but has a unit norm, $\lVert \mathbf{u} \rVert = 1$. This is given by

$$\mathbf{u} = \frac{\mathbf{v}}{\lVert \mathbf{v} \rVert},$$

which is interpreted as vector $\mathbf{v}$ multiplied by a scalar λ being the reciprocal of its norm: $\lambda = 1/\lVert \mathbf{v} \rVert$.

According to the *Cauchy-Schwarz inequality*, for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ inequality $\lVert \mathbf{u} \rVert_2^2 \lVert \mathbf{v} \rVert_2^2 \ge \lVert \mathbf{u}^T \mathbf{v} \rVert_2^2$ holds (proof omitted here). Therefore,

$$-\lVert \mathbf{u} \rVert_2 \lVert \mathbf{v} \rVert_2 \le \mathbf{u}^T \mathbf{v} \le \lVert \mathbf{u} \rVert_2 \lVert \mathbf{v} \rVert_2. \tag{14}$$

If $\mathbf{u}$ and $\mathbf{v}$ are nonzero vectors we can divide (14) by $\lVert \mathbf{u} \rVert_2 \lVert \mathbf{v} \rVert_2$ and obtain

$$-1 \le \frac{\mathbf{u}^T \mathbf{v}}{\lVert \mathbf{u} \rVert_2 \lVert \mathbf{v} \rVert_2} \le 1 \tag{15}$$

The standard formula for the cosine of the angle ϕ between two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ is

$$\cos \phi = \frac{\mathbf{u}^T \mathbf{v}}{\lVert \mathbf{u} \rVert_2 \lVert \mathbf{v} \rVert_2}. \tag{16}$$

From (15) it is obvious that (16) satisfies $|\cos\phi| \leq 1$, i.e., this definition is meaningful. When two vectors $\mathbf{u}$ and $\mathbf{v}$ intersect at right angle the cosine is zero because $\mathbf{u}^T\mathbf{v} = 0$ which corresponds to $\phi = \pi/2$.

3.2.2 Matrix norms

A *matrix norm* is denoted by $\|\cdot\|$ and satisfies the same three properties as a vector norm:

(i) for any nonzero $\mathbf{A}$, $\|\mathbf{A}\| > 0$,

(ii) for any scalar λ, $\|\lambda\mathbf{A}\| = |\lambda|\,\|\mathbf{A}\|$

(iii) for any two matrices $\mathbf{A}$ and $\mathbf{B}$, the *triangle inequality* holds: $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.

Additionally, it is useful for a matrix norm to satisfy the *consistency* property:

(iv) $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\,\|\mathbf{B}\|$.

The three quoted vector norms have their subordinate matrix norm in the following way:

$$\boxed{\begin{array}{ll} \|\mathbf{A}\|_1 = \max\limits_j \|\mathbf{a}_j\|_1 & \text{the maximum absolute column sum,} \\[2ex] \|\mathbf{A}\|_2 = \sigma_1(\mathbf{A}) & \text{the largest singular value (discussion to come),} \\[2ex] \|\mathbf{A}\|_\infty = \max\limits_i \|\mathbf{a}^i\|_1 & \text{the maximum absolute row sum.} \end{array}}$$

A matrix norm and a vector norm are said to be *compatible* or *consistent* if

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\,\|\mathbf{x}\|$$

for every $\mathbf{A}$ and $\mathbf{x}$. A vector norm and its subordinate matrix norm always satisfy this condition.

There is one more important matrix norm, called *Frobenius norm* which is defined by

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2}.$$

The Frobenius norm is compatible with the ℓ_2 (Euclidean) vector norm:

$$\|\mathbf{Ax}\|_2 \leq \|\mathbf{A}\|_F\|\mathbf{x}\|_2.$$

3.3 Vector Spaces

3.3.1 Linear dependence and independence

Linear dependence of m dimensional vectors is defined in the following way: Let

$$\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\} \tag{17}$$

a set of m-vectors. They are said to be *linearly dependent* if the zero vector can be written as a non-trivial linear combination of them, i.e.,

$$\lambda_1\mathbf{a}_1 + \lambda_2\mathbf{a}_2 + \cdots + \lambda_n\mathbf{a}_n = \mathbf{0} \text{ can be achieved with } \lambda_j \neq 0 \text{ for at least one } j.$$

Vectors in (17) are said to be *linearly independent* if the zero vector can be written only as a trivial linear combination of them, i.e.,

$$\lambda_1\mathbf{a}_1 + \lambda_2\mathbf{a}_2 + \cdots + \lambda_n\mathbf{a}_n = \mathbf{0} \text{ implies that } \lambda_j = 0,\ j = 1, 2, \ldots, n.$$

Example 21 *The following three vectors*

$$\mathbf{a}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \ \mathbf{a}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \ \text{ and } \ \mathbf{a}_3 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

are linearly dependent because with $\lambda_1 = -2$, $\lambda_2 = 2$ and $\lambda_3 = -1$ the zero vector can be written in a non-trivial way:

$$-2\begin{bmatrix} -1 \\ 1 \end{bmatrix} + 2\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

However,

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \text{ and } \ \mathbf{a}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

are linearly independent because

$$\lambda_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

implies $\lambda_1 = \lambda_2 = 0$.

The maximum number of linearly independent m-vectors is m since any m dimensional vector can be written as a linear combination of the m dimensional unit vectors (there are m of them). Any set of vectors containing the zero vector is linearly dependent (why?).

If we consider vectors in (17) as columns of matrix $\mathbf{A}$ then the condition of linear dependence of the column vectors can be written as

$$\mathbf{Az} = \mathbf{0} \ \text{ holds with some } \ \mathbf{z} \neq \mathbf{0}.$$

Similarly, linear independence of the columns of $\mathbf{A}$ is equivalent to the condition

$$\mathbf{Az} = \mathbf{0} \ \text{ implies } \ \mathbf{z} = \mathbf{0}. \tag{18}$$

One of the central problems of computational linear algebra is the solution of a set of linear equations of the form

$$\mathbf{Ax} = \mathbf{b}, \tag{19}$$

that is, to find an $\mathbf{x}$ that satisfies equation (19). Since this equation can also be written as

$$x_1\mathbf{a}_1 + \cdots + x_n\mathbf{a}_n = \mathbf{b},$$

the problem can be thought of as looking for a linear combination of the column vectors of $\mathbf{A}$ that gives $\mathbf{b}$. If such an $\mathbf{x}$ can be found then this is a solution to (19) and it is said that the system (19) is *compatible*, otherwise it is *incompatible*.

We can investigate (19) with respect to the relationship among the columns of $\mathbf{A}$. First, assume the columns of $\mathbf{A}$ are linearly dependent, that is $\mathbf{Az} = \mathbf{0}$ with $\mathbf{z} \neq \mathbf{0}$. If (19) is compatible then $\mathbf{x} + \lambda\mathbf{z}$ is also a solution to it with arbitrary λ because

$$\mathbf{A}(\mathbf{x} + \lambda\mathbf{z}) = \mathbf{Ax} + \lambda\mathbf{Az} = \mathbf{b}.$$

This means that if the columns of $\mathbf{A}$ are linearly dependent and the system (19) has a solution then it has infinitely many solutions in the form of

$$\mathbf{x} + \lambda\mathbf{z}. \tag{20}$$

Example 22 *Let*

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 4 \\ 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

Columns of $\mathbf{A}$ *are linearly dependent and* $\mathbf{Ax} = \mathbf{b}$ *has a solution because with* $\mathbf{z} = [2, -2, 1]^T$ *and* $\mathbf{x} = [3, 1, 1]^T$

$$\mathbf{Az} = \begin{bmatrix} -1 & 1 & 4 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \textit{and} \quad \mathbf{Ax} = \begin{bmatrix} -1 & 1 & 4 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

Using (20), all solutions can be written in the form of

$$\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix} \quad \textit{with arbitrary } \lambda. \textit{ For instance, } \lambda = 1 \textit{ gives} \quad \begin{bmatrix} 5 \\ -1 \\ 2 \end{bmatrix}.$$

It is easy to verify that this is also a solution to the system.

If the columns of $\mathbf{A}$ are linearly independent and $\mathbf{Ax} = \mathbf{b}$ has a solution then this solution is unique. To prove it, assume that $\mathbf{x}$ is not unique and there exists an $\hat{\mathbf{x}}$ such that $\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$. Combining it with $\mathbf{Ax} = \mathbf{b}$, we obtain $\mathbf{A}(\mathbf{x} - \hat{\mathbf{x}}) = \mathbf{0}$. However, because of the linear independence of the columns of $\mathbf{A}$ as in (18), $\mathbf{x} - \hat{\mathbf{x}} = \mathbf{0}$, that is $\mathbf{x} = \hat{\mathbf{x}}$ which means that the solution is unique.

During the proof we did not assume that $\mathbf{A}$ was square. It means that the uniqueness holds even if the number of columns in $\mathbf{A}$ is less than the dimension of the column vectors (the number of rows in $\mathbf{A}$), that is, $n < m$.

Example 23 *Let*

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}, \quad \textit{and} \quad \mathbf{b} = \begin{bmatrix} 3 \\ 1 \\ -1 \end{bmatrix}.$$

The columns of $\mathbf{A}$ *are linearly independent and* $\mathbf{x} = [1, 2]^T$ *is a solution to* $\mathbf{Ax} = \mathbf{b}$ *because*

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -1 \end{bmatrix}.$$

From the observation above, we know that this is 'the' solution, no other solution can exist.

This example shows that $\mathbf{b}$ is a linear combination of the columns of $\mathbf{A}$.

The last possibility, regarding the relationship between $\mathbf{A}$ and $\mathbf{b}$, is that $\mathbf{b}$ cannot be written as a linear combination of the columns of $\mathbf{A}$, that is, there is no $\mathbf{x}$ that satisfies $\mathbf{Ax} = \mathbf{b}$ regardless of the linear dependence or independence of the columns of $\mathbf{A}$. The following example demonstrates this situation.

Example 24 *If*

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}, \quad \textit{and} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

then there is no $\mathbf{x} = [x_1, x_2]^T$ *that satisfies*

$$x_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

The summary of the main points of this subsection is the following:

1. If the columns of $\mathbf{A}$ are linearly dependent and the linear system $\mathbf{Ax} = \mathbf{b}$ has a solution, it has an infinite number of solutions.

2. If the columns of $\mathbf{A}$ are linearly dependent, the linear system $\mathbf{Ax} = \mathbf{0}$ always has a solution with $\mathbf{x} \neq \mathbf{0}$.

3. If the columns of $\mathbf{A}$ are linearly independent and $\mathbf{Ax} = \mathbf{b}$ has a solution, the solution is unique.

3.3.2 Range and null spaces

This subsection gives a more formal framework for the characterization of the solution of a set of linear equations. First, we define the notion of a *subspace*.

Let S be a set containing m dimensional vectors. S is a subspace of $\mathbb{R}^m$ if for scalars λ_1 and λ_2 and vectors $\mathbf{a}_1, \mathbf{a}_2 \in S$, any linear combination is also in S:

$$\lambda_1,\ \lambda_2 \in \mathbb{R} \ \text{ and } \ \mathbf{a}_1, \mathbf{a}_2 \in S \ \text{ imply } \ \lambda_1\mathbf{a}_1 + \lambda_2\mathbf{a}_2 \in S.$$

A direct consequence of this property is that every subspace contains the zero vector (why?).

Given a subspace S, a set of vectors $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ is said to *span* S if every vector $\mathbf{x} \in S$ can be written as a linear combination of these vectors:

$$\mathbf{x} = \lambda_1\mathbf{a}_1 + \cdots + \lambda_k\mathbf{a}_k.$$

The *trivial subspace* contains only the zero vector. For any nontrivial subspace, there is a smallest positive integer r such that every vector in the subspace can be expressed as a linear combination of a fixed set of r vectors. This number r is called the *dimension* or *rank* of the subspace. Any such set of r vectors is called a *basis* for the subspace. There can be several different bases for the same subspace. The vectors in a basis are linearly independent (otherwise at least one of them could be expressed as a linear combination of the others and in this way r would not be minimal). For any positive integer k, a set of k linearly independent vectors is a basis for a k-dimensional subspace.

The set of all m-vectors that are linear combinations of the columns of the $m \times n$ matrix $\mathbf{A}$ is called the *range space, column space* or the *range* of $\mathbf{A}$ and is denoted by range($\mathbf{A}$).

A vector $\mathbf{b}$ is in the range of $\mathbf{A}$ if and only if there exists a linear combination of the column vectors of $\mathbf{A}$ that gives $\mathbf{b}$, that is, $\mathbf{b} \in \text{range}(\mathbf{A})$ *iff* $\exists \mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{b} = \mathbf{Ax}$. The system $\mathbf{Ax} = \mathbf{b}$ is thus compatible if and only if $\mathbf{b}$ is in the range of $\mathbf{A}$. The range of $\mathbf{A}$ is a subspace of $\mathbb{R}^m$ (why?).

Example 25 *For the matrix* $\mathbf{A}$ *of Example 23* range($\mathbf{A}$) *consists of all vectors of the form:*

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \beta_1 + \beta_2 \\ \beta_1 \\ \beta_1 - \beta_2 \end{bmatrix} \tag{21}$$

with arbitrary scalars β_1 *and* β_2.

For example, vectors $\mathbf{b}_1 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$ *and* $\mathbf{b}_4 = \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix}$ *lie in* range($\mathbf{A}$) *because*

$$\begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \end{bmatrix}.$$

At the same time, vector $\mathbf{b} = [1, 2, 1]^T$ *of Example 24 does not lie in* range($\mathbf{A}$) *because it cannot be written in the form (21).*

The *row space* of $\mathbf{A}$ (also called the range of $\mathbf{A}^T$) is defined in a similar fashion. It is the set of all n-vectors that are linear combinations of the row vectors of $\mathbf{A}$. This is a subspace of the $\mathbb{R}^n$ and is denoted by range($\mathbf{A}^T$). Formally, $\mathbf{x} \in \text{range}(\mathbf{A}^T)$ *iff* $\exists \mathbf{v}$ such that $\mathbf{x} = \mathbf{A}^T\mathbf{v}$.

The *column rank* of a matrix $\mathbf{A}$ (and hence the dimension of range($\mathbf{A}$)) is the maximum number of linearly independent columns in $\mathbf{A}$. Similarly, the *row rank* of $\mathbf{A}$ is the maximum number of linearly independent rows. It can be shown that the row and column ranks of a given matrix $\mathbf{A}$ are equal; their common value is called the *rank of the matrix* and is denoted by rank($\mathbf{A}$). For obvious reasons, $\text{rank}(\mathbf{A}) \leq \min\{m, n\}$.

A matrix whose columns are linearly independent is said to have *full column rank*. The definition of *full row rank* is similar. The matrix $\mathbf{A}$ is said to have *full rank* if it has either full column rank or full row rank. In case of full rank, $\text{rank}(\mathbf{A}) = \min\{m, n\}$. If $\text{rank}(\mathbf{A}) < \min\{m, n\}$ then the matrix is said to be *rank-deficient*.

Example 26 *Consider the following matrices:*

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

Obviously, $\text{rank}(\mathbf{A}) = 0$. *The columns of* $\mathbf{B}$ *are linearly dependent (verify!) with* $\text{rank}(\mathbf{B}) = 1$. *It means that* $\mathbf{B}$ *is rank-deficient because in this case* $\min\{m, n\} = 2$. *(Note that rows of* $\mathbf{B}$ *are also linearly dependent.) As rows 1 and 2 of matrix* $\mathbf{C}$ *are linearly independent it has full row rank and, consequently,* $\text{rank}(\mathbf{C}) = 2$.

Any $m \times n$ matrix $\mathbf{A}$ defines two other important subspaces apart from range($\mathbf{A}$) and range($\mathbf{A}^T$). The set of all m-vectors *orthogonal* to column vectors of $\mathbf{A}$ (and hence to range($\mathbf{A}$)) is called the *null space* of $\mathbf{A}^T$ and denoted by null($\mathbf{A}^T$). Formally, the elements of null($\mathbf{A}^T$) are defined as follows:

$$\mathbf{z} \in \text{null}(\mathbf{A}^T) \ \textit{iff} \ \mathbf{A}^T\mathbf{z} = \mathbf{0}.$$

It can be shown that null($\mathbf{A}^T$) is a subspace (show it, use definition of null space and linearity of operations with $\mathbf{A}^T$).

The intersection of the two subspaces range($\mathbf{A}$) and null($\mathbf{A}^T$) is the zero vector. Any m-vector can be written as a linear combination of vectors in range($\mathbf{A}$) and null($\mathbf{A}^T$). These two subspaces are *orthogonal complements* of one another. The sum of the dimensions of range($\mathbf{A}$) and null($\mathbf{A}^T$) is m.

Analogous properties hold for range($\mathbf{A}^T$) and null($\mathbf{A}$). Any n-vector can be written as a linear combination of vectors in range($\mathbf{A}^T$) and null($\mathbf{A}$). The sum of the dimensions of range($\mathbf{A}^T$) and null($\mathbf{A}$) is n.

Example 27 *Any vector in the null space of matrix* $\mathbf{A}^T$ *of Example 23 satisfies*

$$\mathbf{A}^T\mathbf{z} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} z_1 + z_2 + z_3 \\ z_1 - z_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which directly yields $z_1 = z_3$ *and, after back-substitution,* $z_2 = -2z_3$. *This allows arbitrary value for* z_3, *say* β_3 *and the rest is determined as* $z_1 = \beta_3$ *and* $z_2 = -2\beta_3$. *It means that any vector* $\mathbf{z}$ *in null(*$\mathbf{A}^T$*) is of the form:*

$$\mathbf{z} = \begin{bmatrix} \beta_3 \\ -2\beta_3 \\ \beta_3 \end{bmatrix}.$$

One such vector is

$$\mathbf{b}_3 = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}. \tag{22}$$

which corresponds to $\beta_3 = -1$. *The dimension of null(*$\mathbf{A}^T$*) is one.*

Because range($\mathbf{A}$) and null($\mathbf{A}^T$) contain only the zero vector in common, the expression of every nonzero m-vector $\mathbf{b}$ as a sum of two vectors is unique:

$$\mathbf{b} = \mathbf{b}_R + \mathbf{b}_N, \text{ where } \mathbf{b}_R \in \text{range}(\mathbf{A}), \ \mathbf{b}_N \in \text{null}(\mathbf{A}^T),$$

and, because of the orthogonality, vectors $\mathbf{b}_R$ and $\mathbf{b}_N$ satisfy

$$\mathbf{b}_R^T\mathbf{b}_N = 0 \text{ and, furthermore, } \mathbf{b}^T\mathbf{b} = \mathbf{b}_R^T\mathbf{b}_R + \mathbf{b}_N^T\mathbf{b}_N.$$

Given $\mathbf{A}$, vectors $\mathbf{b}_R$ and $\mathbf{b}_N$ are the range-space and null-space components of vector $\mathbf{b}$.

Similar things can be said in the range space of $\mathbf{A}^T$, namely, any nonzero n-vector $\mathbf{x}$ has a unique representation in the form of

$$\mathbf{x} = \mathbf{x}_R + \mathbf{x}_N, \text{ where } \mathbf{x}_R \in \text{range}(\mathbf{A}^T) \text{ and } \mathbf{x}_N \in \text{null}(\mathbf{A}).$$

Example 28 *Let* $\mathbf{A}$ *be the matrix of Example 23*

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}, \quad \textit{and} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix}.$$

Using vector $\mathbf{b}_1 = [2, 1, 0]^T$ *of Example 25 and* $\mathbf{b}_3 = [-1, 2, -1]^T$ *of (22) we can give the unique representation of* $\mathbf{b}$ *in terms of its range(*$\mathbf{A}$*) and null(*$\mathbf{A}^T$*) components as*

$$\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_3 = \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix},$$

where $\mathbf{b}_1 \in \text{range}(\mathbf{A})$ *and* $\mathbf{b}_3 \in$ *null(*$\mathbf{A}^T$*), that is, in this break-down* $\mathbf{b}_R = \mathbf{b}_1$ *and* $\mathbf{b}_N = \mathbf{b}_3$*.*

Additionally, $\mathbf{b}_1^T\mathbf{b}_3 = 2 \times (-1) + 1 \times 2 + 0 \times (-1) = 0$, *and* $\mathbf{b}^T\mathbf{b} = 11 = \mathbf{b}_1^T\mathbf{b}_1 + \mathbf{b}_3^T\mathbf{b}_3$*.*

3.3.3 Singular and nonsingular matrices

For the characterization of linear dependence and independence of square matrices a special terminology is used. In this subsection $\mathbf{A}$ will denote an $m \times m$ (square) matrix.

A square matrix with linearly independent columns is said to be *nonsingular*. A square matrix whose columns are linearly dependent is said to be *singular*. If $\mathbf{A}$ is nonsingular, its columns (rows) span the m-dimensional space $\mathbb{R}^m$. This implies that the subspaces null($\mathbf{A}^T$) and null($\mathbf{A}$) contain only the zero vector.

The following properties follow immediately from the previous general discussion of linear independence:

1. $\mathbf{A}$ is nonsingular if $\mathbf{A}\mathbf{x} = \mathbf{0}$ only when $\mathbf{x} = \mathbf{0}$.

2. If $\mathbf{A}$ is nonsingular, $\mathbf{A}\mathbf{x} = \mathbf{b}$ always has a unique solution.

3. $\mathbf{A}$ is singular iff $\exists \mathbf{x} \neq \mathbf{0}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0}$.

4. If $\mathbf{A}$ is singular and $\mathbf{A}\mathbf{x} = \mathbf{b}$ has a solution, it has infinitely many solutions.

For every nonsingular matrix $\mathbf{A}$ there exists an associated matrix $\mathbf{A}^{-1}$, called $\mathbf{A}$ *inverse* or inverse of $\mathbf{A}$, such that

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

Such an $\mathbf{A}^{-1}$ is unique, nonsingular and satisfies the property that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. If $\mathbf{A}^{-1}$ exists, it is sometimes said that $\mathbf{A}$ is invertible. The inverse of the inverse is the original matrix: $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$.

If $\mathbf{A}$ and $\mathbf{B}$ are nonsingular, the product $\mathbf{AB}$ is also nonsingular (why?) and

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}.$$

If $\mathbf{A}$ is nonsingular, $\mathbf{A}^T$ is also nonsingular (why?) and

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T, \quad \text{which is denoted also by } \mathbf{A}^{-T}.$$

Computationally it is demanding to compute the inverse of a general matrix. However, there are some points to be noted here. First, usually there is no need to explicitly determine the elements of the inverse of a general matrix. Second, certain kinds of matrices have easily obtainable inverses. If $\mathbf{D}$ is a nonsingular diagonal matrix, $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_m)$ then $\mathbf{D}^{-1}$ is also a diagonal matrix:

$$\mathbf{D}^{-1} = \operatorname{diag}(d_1^{-1}, \ldots, d_m^{-1}).$$

A square diagonal matrix is singular iff at least one of its diagonal elements is zero (why?). Third, if $\mathbf{Q}$ is orthogonal, then $\mathbf{QQ}^T = \mathbf{I}$. It means that $\mathbf{Q}^T$ plays the role of the inverse and, in fact, it is the inverse of $\mathbf{Q}$:

$$\mathbf{Q}^{-1} = \mathbf{Q}^T.$$

Hence, if we know that a matrix is orthogonal, its inverse is available without any computations.

Fourth, the inverse of an elementary matrix $\mathbf{E} = \mathbf{I} - \alpha\mathbf{uv}^T$ is also an elementary matrix involving the same two vectors as $\mathbf{E}$ itself. If $\alpha\mathbf{u}^T\mathbf{v} \neq 1$ then

$$\mathbf{E}^{-1} = \left(\mathbf{I} - \alpha\mathbf{uv}^T\right)^{-1} = \mathbf{I} - \beta\mathbf{uv}^T, \quad \text{where } \beta = \frac{\alpha}{\alpha\mathbf{u}^T\mathbf{v} - 1}.$$

3.4 Eigenvalues, Eigenvectors, Singular Values

3.4.1 Eigenvalues and eigenvectors

For any square matrix $\mathbf{A}$, there exists at least one number λ and an associated nonzero vector $\mathbf{u}$ such that

$$\mathbf{Au} = \lambda\mathbf{u}. \tag{23}$$

Verbally, the transformation of $\mathbf{u}$ by $\mathbf{A}$ does not change the direction of $\mathbf{u}$. In the general case it is possible that such a λ is a complex number. A value of λ for which (23) holds is called an *eigenvalue* of $\mathbf{A}$ and the corresponding vector is called an *eigenvector*.

Example 29 *Let*

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}, \quad \mathbf{u}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Now,

$$\mathbf{Au}_1 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3\mathbf{u}_1,$$

that is, $\lambda_1 = 3$ is an eigenvalue and $\mathbf{u}_1 = [1, 1]^T$ is the corresponding eigenvector of $\mathbf{A}$. Furthermore, we have

$$\mathbf{Bv} = \begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \end{bmatrix} = 3\mathbf{v},$$

so, $\lambda = 3$ is an eigenvalue and $\mathbf{v} = [1, 0]^T$ is the corresponding eigenvector of $\mathbf{B}$. Note that $\lambda = 3$ is an eigenvalue for both matrices but the eigenvectors are different.

From (23), which can be written as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{u} = \mathbf{0}$, it follows that for λ to be an eigenvalue it is necessary and sufficient that $\mathbf{A} - \lambda\mathbf{I}$ is singular.

Any $m \times m$ matrix $\mathbf{A}$ has m eigenvalues $\lambda_1, \dots, \lambda_m$. These eigenvalues are not necessarily distinct and can be complex numbers. They are the roots of the m-th degree polynomial equation $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. Here, 'det' denotes the determinant (not discussed in these notes). Sometimes the set of eigenvalues of $\mathbf{A}$ is denoted by $\{\lambda_1(\mathbf{A}), \dots, \lambda_m(\mathbf{A})\}$.

In Example 29, $\mathbf{A}$ has two eigenvalues, $\lambda_1 = 3$ and $\lambda_2 = 1$, while the corresponding eigenvectors are $\mathbf{u}_1 = [1, 1]^T$ and $\mathbf{u}_2 = [-1, 1]^T$. At the same time, $\mathbf{B}$ has a double eigenvalue $\lambda = \lambda_1 = \lambda_2 = 3$ (the *multiplicity* of this eigenvalue is two) and both have the same eigenvector $\mathbf{v} = [1, 0]^T$.

Any nonzero multiple of an eigenvector is also an eigenvector, since the linearity of the transformation ensures that $\mathbf{A}(\gamma\mathbf{u}) = \lambda(\gamma\mathbf{u})$ for any nonzero scalar γ. This makes it possible to implicitly assume that the eigenvectors have a Euclidean (ℓ_2) norm of one. The eigenvalues and eigenvectors together comprise the *eigensystem* of a matrix.

The sum of the diagonal elements of a square matrix $\mathbf{A}$ (called the *trace* of $\mathbf{A}$) is equal to the sum of the eigenvalues:

$$\operatorname{trace}(\mathbf{A}) = \sum_{i=1}^{m} a_{ii} = \sum_{i=1}^{m} \lambda_i(\mathbf{A}).$$

As an illustration of it, check $\mathbf{A}$ of Example 29, where $a_{11} + a_{22} = 2 + 2 = 4$ and $\lambda_1 + \lambda_2 = 3 + 1 = 4$.

The product of the eigenvalues is equal to the determinant of $\mathbf{A}$:

$$\det(\mathbf{A}) = \prod_{i=1}^{m} \lambda_i(\mathbf{A}).$$

For $\mathbf{A}$ of Example 29, $\det(\mathbf{A}) = 2 \times 2 - 1 \times 1 = 3$, and $\lambda_1 \times \lambda_2 = 3 \times 1 = 3$.

3.4.2 Spectral decomposition

If we multiply both sides of (23) by a nonsingular matrix $\mathbf{S}$, we obtain $\mathbf{SAu} = \lambda(\mathbf{Su})$. Using $\mathbf{S}^{-1}\mathbf{S} = \mathbf{I}$, it follows that

$$\mathbf{SAu} = \mathbf{SAS}^{-1}\mathbf{Su} = \mathbf{SAS}^{-1}(\mathbf{Su}) = \lambda(\mathbf{Su}), \tag{24}$$

which shows that λ is an eigenvalue and $\mathbf{Su}$ is an eigenvector of $\mathbf{SAS}^{-1}$. Two matrices are said to be *similar* if they have the same eigenvalues. (24) verifies that $\mathbf{SAS}^{-1}$ is similar to $\mathbf{A}$. The matrix $\mathbf{SAS}^{-1}$ is called a *similarity transform* of $\mathbf{A}$.

If $\mathbf{A}$ is singular, there exists a nonzero vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{0}$, which shows that a singular matrix has at least one zero eigenvalue. If $\mathbf{A}$ is nonsingular, all its eigenvalues are nonzero, and the eigenvalues of $\mathbf{A}^{-1}$ are the reciprocals of the eigenvalues of $\mathbf{A}$.

The eigensystem of a nonsingular symmetric matrix $\mathbf{A}$ has two special properties:

1. All eigenvalues of $\mathbf{A}$ are real.

2. $\mathbf{A}$ has m orthogonal eigenvectors which form a basis for $\mathbb{R}^m$. They may be normalized so that $\|\mathbf{u}_i\|_2 = 1,\ i = 1, \dots, m$ (that is, $\mathbf{u}_i^T\mathbf{u}_i = 1$ and $\mathbf{u}_i^T\mathbf{u}_j = 0$ for $i \neq j$).

Let $\mathbf{Q} = [\mathbf{u}_1, \dots, \mathbf{u}_m]$. Since the column vectors are normalized and are orthogonal, $\mathbf{Q}$ is orthogonal and $\mathbf{Q}^T = \mathbf{Q}^{-1}$. It follows that with this $\mathbf{Q}$

$$\mathbf{Q}^{-1}\mathbf{AQ} = \mathbf{Q}^T\mathbf{AQ} = \mathbf{Q}^T[\mathbf{Au}_1, \dots, \mathbf{Au}_m] = \mathbf{Q}^T[\lambda_1\mathbf{u}_1, \dots, \lambda_m\mathbf{u}_m].$$

Comparing the first part with the last one:

$$\mathbf{Q}^{-1}\mathbf{AQ} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix} = \mathbf{\Lambda} \tag{25}$$

and therefore a symmetric $\mathbf{A}$ is similar to a diagonal matrix $\mathbf{\Lambda} = \mathrm{diag}(\lambda_i)$, where λ_i's are the eigenvalues of $\mathbf{A}$.

Premultiplying (25) by $\mathbf{Q}$ and postmultiplying by $\mathbf{Q}^{-1}$, we obtain the *spectral decomposition* of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}.$$

Recall that $\mathbf{x}^T$ is a row vector, so is $\mathbf{x}^T\mathbf{A}$. Therefore, $\mathbf{x}^T\mathbf{A}\mathbf{x} = (\mathbf{x}^T\mathbf{A})\mathbf{x}$ is a dot product resulting in a number. A symmetric matrix $\mathbf{A}$ is said to be *positive definite* if

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \quad \text{for all nonzero } \mathbf{x}. \tag{26}$$

If the relation is '$\geq$' then $\mathbf{A}$ is said to be *positive semidefinite*. *Negative definite* and *negative semidefinite* are defined with relations '$<$' and '$\leq$' in (26). The matrix $\mathbf{A}$ is *indefinite* if $\mathbf{x}^T\mathbf{A}\mathbf{x}$ is positive for some $\mathbf{x}$ and negative for others.

It is easy to obtain a connection between definiteness and eigenvalues of $\mathbf{A}$. For any $\mathbf{x}$ let $\mathbf{y} = \mathbf{Q}^{-1}\mathbf{x}$ ($\Rightarrow$ $\mathbf{x} = \mathbf{Q}\mathbf{y}$, $\mathbf{x}^T = \mathbf{y}^T\mathbf{Q}^T$), where $\mathbf{Q}$ is defined as above (remember: $\mathbf{Q}^T = \mathbf{Q}^{-1}$). Then $\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{y}^T\mathbf{Q}^T\mathbf{A}\mathbf{Q}\mathbf{y} = \mathbf{y}^T(\mathbf{Q}^T\mathbf{A}\mathbf{Q})\mathbf{y} = \mathbf{y}^T\mathbf{\Lambda}\mathbf{y}$ which implies that

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \sum_{i=1}^{m} \lambda_i y_i^2.$$

Since y_i's are arbitrary (because $\mathbf{x}$ is), it is clear that $\mathbf{A}$ is positive definite (or positive semidefinite) if and only if all λ_i eigenvalues of $\mathbf{A}$ are positive (or nonnegative).

3.4.3 Singular value decomposition

Any $m \times n$ matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{27}$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix, $\mathbf{V}$ is an $n \times n$ orthogonal matrix, and $\mathbf{S}$ is an $m \times n$ diagonal (!) matrix, $\mathbf{S} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)$, with $p = \min\{m, n\}$ and $\sigma_i \geq 0,\ i = 1, \ldots, p$. The nonnegative numbers $\{\sigma_i\}$ are called the *singular values* of $\mathbf{A}$, and (27) is called the *singular value decomposition* (SVD). The SVD is the most informative general representation of a matrix. The convention is usually adopted that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$, so that $\sigma_1(\mathbf{A})$ denotes the largest singular value of $\mathbf{A}$.

If $\mathbf{A}$ has rank r and $r > 0$, then $\mathbf{A}$ has exactly r strictly positive singular values, so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_p = 0$. If $\mathbf{A}$ has full rank, all its singular values are nonzero (that is, positive). The singular values of $\mathbf{A}$ are the square roots of the eigenvalues of the symmetric matrix $\mathbf{A}^T\mathbf{A}$ (if $m \geq n$) or of the symmetric matrix $\mathbf{A}\mathbf{A}^T$ (if $m < n$). If $\mathbf{A}$ is symmetric, its singular values are the absolute values of its eigenvalues.

3.5 Solution of Systems of Linear Equations

The fundamental problem of this section is the solution of the set of linear equations given in the following form:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{28}$$

where vector $\mathbf{x}$ is to be determined. We already know that a solution exists only if (28) is compatible, i.e., $\mathbf{b}$ lies in the range of $\mathbf{A}$.

3.5.1 Triangular systems

If **A** is upper triangular, the solution of (28) can easily be computed. Consider the following example:

$$\begin{array}{rcrcrcr} x_1 & + & 2x_2 & + & 3x_3 & = & 5 \\ & & -x_2 & + & x_3 & = & 3 \\ & & & & -3x_3 & = & -6 \end{array} \tag{29}$$

From the third equation x_3 can be determined as $x_3 = (-6)/(-3) = 2$. However, knowing that $x_3 = 2$, it can be substituted into equation 2 where x_2 remains the only unknown:

$$-x_2 + 2 = 3,$$

which immediately gives $x_2 = -1$. Now, we know the values of x_2 and x_3 and can substitute them into equation 1:

$$x_1 + 2(-1) + 3(2) = 5,$$

from which $x_1 = 1$ follows. This completes the solution procedure. The solution of (29) is $\mathbf{x} = [1, -1, 2]^T$.

What we have just done is called *back-substitution.* Thus, we can say that if **A** of (28) is upper triangular, (28) can be solved by back-substitution. In general, if **A** is an $m \times m$ nonsingular upper triangular matrix then the diagonal elements are all nonzero and the solution, which is unique, can be computed by the following formula:

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^{m} a_{ij} x_j \right), \quad \text{for} \quad i = m, m-1, \ldots, 1,$$

where the sum is defined to be zero for $i = m$. The formula reflects that at the time of calculating x_i all variables $x_{i+1}, \ldots, x_m$ are known.

We can conclude that the case when **A** is upper triangular is a favourable one that can be solved in a rather straightforward way. Let us now consider an example when **A** is lower triangular:

$$\begin{array}{rcrcrcr} 2x_1 & & & & & = & 4 \\ x_1 & + & x_2 & & & = & 1 \\ x_1 & - & 3x_2 & + & 2x_3 & = & -1 \end{array}$$

From the first equation we immediately get $x_1 = 4/2 = 2$. Substituting it into the second equation: $2 + x_2 = 1$ from which $x_2 = -1$ follows. Now, the values of x_1 and x_2 can be substituted into equation 3 and we obtain $2 - 3(-1) + 2x_3 = -1$. Solving it for x_3, we get $x_3 = (-6)/2 = -3$. Thus, the solution is $\mathbf{x} = [2, -1, -3]^T$.

The systematic way the solution was obtained is valid for the general case and is called *forward substitution.* Formally, if **A** of (28) is a nonsingular lower triangular matrix then the solution to (28) is unique and can be computed by the following formula:

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j \right), \quad \text{for} \quad i = 1, 2, \ldots, m,$$

where the sum is defined to be zero for $i = 1$. The formula shows that at the beginning of step i variables $x_1, \ldots, x_{i-1}$ are known.

Thus, we can conclude that if **A** is upper or lower triangular, (28) can be solved easily. This observation suggests that one should try to reduce the general case to one of the triangular forms to take advantage of the solution procedures just developed.

These advantages of the triangular cases have motivated the development of the *LU factorization* (also called *LU decomposition*) of a nonsingular matrix $\mathbf{A}$. If

$$\mathbf{A} = \mathbf{LU},$$

then (28) can be written as

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}. \tag{30}$$

Defining $\mathbf{y} = \mathbf{Ux}$, from (30) we get $\mathbf{Ly} = \mathbf{b}$. This latter is a lower triangular system that can be solved for $\mathbf{y}$ by forward substitution. When $\mathbf{y}$ becomes known, $\mathbf{Ux} = \mathbf{y}$ can be solved for $\mathbf{x}$ by back-substitution. The *LU* form of $\mathbf{A}$ can practically be obtained by the Gaussian elimination discussed in the next section.

3.5.2 Gaussian elimination

The Gaussian elimination, GE, (named after the German mathematician Karl Friedrich Gauss [1777–1855]) is the basic computational technique to solve (28). Its application can also reveal that no solution exists or there are infinitely many solutions.

GE performs transformations on the linear system to be solved in such a way that the transformed system is equivalent to the original one in the sense that both have the same solution. Therefore, if a solution to the transformed system is obtained, it is also the solution to the original problem. The transformations will result in an upper triangular system that can be solved by back-substitution as discussed in section 3.5.1.

To facilitate GE, we introduce elementary row operations (ERO) on (28) that are equivalent transformations in the above sense. The notion *row* now refers to an equation of (28): $\mathbf{a}^i\mathbf{x} = b_i$. To illustrate the operations, we use the following example:

Example 30

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & - & x_3 & = & 2 \\ 2x_1 & - & x_2 & + & 4x_3 & = & 4 \\ 3x_1 & - & 2x_2 & - & x_3 & = & -2 \end{array} \tag{31}$$

The matrix of the coefficients (including $\mathbf{b}$ *as the last column):*

$$\mathbf{A}^* = \left[\begin{array}{rrr|r} 1 & 1 & -1 & 2 \\ 2 & -1 & 4 & 4 \\ 3 & -2 & -1 & -2 \end{array}\right]$$

ERO-1: Multiplication of a row by a constant. Mathematical form: $\lambda\mathbf{a}^i\mathbf{x} = \lambda b_i$. In pseudo-code form: `row(i):=lambda * row(i)`, with obvious meaning of `row(i)`.

Example 31 *Multiply row 1 of* $\mathbf{A}^*$ *by 2:*

$$\left[\begin{array}{rrr|r} 2 & 2 & -2 & 4 \\ 2 & -1 & 4 & 4 \\ 3 & -2 & -1 & -2 \end{array}\right]$$

ERO-2: Replacing a row by subtracting from it a scalar multiple of another row. Mathematically, row i becomes $[\mathbf{a}^i - \lambda\mathbf{a}^k, b_i - \lambda b_k]$ if the index of the other row is k. In pseudo-code: `row(i):=row(i) - lambda * row(k)`.

Example 32 *Consider* $\mathbf{A}^*$ *and multiply row 1 by 3 and subtract it from row 3:*

$$\left[\begin{array}{rrr|r} 1 & 1 & -1 & 2 \\ 2 & -1 & 4 & 4 \\ 0 & -5 & 2 & -8 \end{array}\right]$$

Note: row 1 remains unaltered.

ERO-3: Swapping rows i and k, pseudo-code: `swap(row(i),row(k))`, with obvious interpretation of function `swap()`.

Example 33 *Swap rows 2 and 3 of* $\mathbf{A}^*$*:*

$$\left[\begin{array}{rrr|r} 1 & 1 & -1 & 2 \\ 3 & -2 & -1 & -2 \\ 2 & -1 & 4 & 4 \end{array}\right]$$

Let us consider the system of equations in (31). If we perform ERO-2 on row 2 with row 1 and $\lambda = 2$ as `row(2):=row(2)-2*row(1)` and then ERO-2 on row 3 with row 1 and $\lambda = 3$ as `row(3):=row(3)-3*row(1)`, the coefficients of x_1 in rows 2 and 3 become zero:

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & - & x_3 & = & 2 \\ & - & 3x_2 & + & 6x_3 & = & 0 \\ & - & 5x_2 & + & 2x_3 & = & -8 \end{array}$$

In other words, the first equation has remained unaltered and we have eliminated variable x_1 from all but the first equation. At this stage we can forget about the first equation and consider the remaining system and do exactly the same as with the original. Now we use equation 2 and eliminate x_2 from all equations (in the example from only one) so that subtract a multiple of $(-5)/(-3)$ of row 2 from row 3 (ERO-2 on row 3): `row(3):=row(3)-(5/3)*row(2)` and obtain:

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & - & x_3 & = & 2 \\ & - & 3x_2 & + & 6x_3 & = & 0 \\ & & & - & 8x_3 & = & -8 \end{array}$$

This is already the well known upper triangular system that can be solved by back-substitution resulting in the solution $\mathbf{x} = [1, 2, 1]^T$.

The solution of a general $m \times m$ system follows the above pattern with some fine tuning. The principle idea is to eliminate one variable at each stage from the remaining equations. This can be achieved by applying EROs in an appropriate manner. After each elimination the remaining part of the system (the matrix of the system) changes. At the beginning of stage i the coefficients will be given a superscript $^{(i)}$ in parentheses. As such, the original system will have superscript $^{(1)}$ as follows:

$$\begin{array}{c} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1m}^{(1)}x_m = b_1^{(1)} \\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + \cdots + a_{2m}^{(1)}x_m = b_2^{(1)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}^{(1)}x_1 + a_{m2}^{(1)}x_2 + \cdots + a_{mm}^{(1)}x_m = b_m^{(1)} \end{array}$$

It is easy to see that the appropriate multiple of row 1 to subtract from row 2 in order to eliminate x_1 from row 2 is $a_{21}^{(1)}/a_{11}^{(1)}$, assuming that $a_{11}^{(1)} \neq 0$. If $a_{11}^{(1)} = 0$, we look for a row k where $a_{k1}^{(1)} \neq 0$ and perform ERO-3 with rows 1 and k. In general, to eliminate x_1 from row i, $a_{i1}^{(1)}/a_{11}^{(1)}$ times row 1 must be subtracted from row i. Performing this operation on all rows $i = 2, \ldots, m$ (i.e., ERO-2 on row i with row 1), the following system is obtained:

$$\begin{array}{ccccccc}
a_{11}^{(1)} x_1 & + & a_{12}^{(1)} x_2 & + \cdots + & a_{1m}^{(1)} x_m & = & b_1^{(1)} \\
 & & a_{22}^{(2)} x_2 & + \cdots + & a_{2m}^{(2)} x_m & = & b_2^{(2)} \\
 & & \vdots & \ddots & \vdots & & \vdots \\
 & & a_{m2}^{(2)} x_2 & + \cdots + & a_{mm}^{(2)} x_m & = & b_m^{(2)}
\end{array}$$

Note superscript $^{(2)}$ in rows $2, \ldots, m$. If all coefficients of x_1 are zero in all rows then the matrix is singular. In this case there are two possibilities: either we have infinitely many solutions or there is no solution at all (see the end of section 3.3.1). To find out what is the case we simply ignore the empty column (where all coefficients are zero) and move over to the column of x_2.

In the next step we consider the reduced problem with coefficients having superscript $^{(2)}$. Again, we look for a row k which has a nonzero coefficient in the column of x_2, that is, $a_{k2}^{(2)} \neq 0$. If such a row cannot be found, the matrix is singular and, again, we move over to the next variable. If one is found we swap it with row 2 making $a_{22}^{(2)} \neq 0$. (If $k = 2$ then formally a `swap(row(2),row(2))` is performed which, of course, involves no operation.) To eliminate x_2 from rows $3, \ldots, m$, the following multiples of row 2 have to be subtracted from rows $3, \ldots, m$: $a_{32}^{(2)}/a_{22}^{(2)}, \ldots, a_{m2}^{(2)}/a_{22}^{(2)}$, respectively. The coefficients of the newly obtained system are superscribed by $^{(3)}$.

In the general step k the following transformations take place:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \left(a_{ik}^{(k)}/a_{kk}^{(k)}\right) a_{kj}^{(k)}, \quad i, j > k \tag{32}$$

and

$$b_i^{(k+1)} = b_i^{(k)} - \left(a_{ik}^{(k)}/a_{kk}^{(k)}\right) b_k^{(k)}, \quad i > k. \tag{33}$$

The denominator in the transformation (in this case $a_{kk}^{(k)}$) is called *pivot element*, while the $a_{ik}^{(k)}/a_{kk}^{(k)}$ ratios are the *multipliers* of the elimination.

Continuing this procedure (if possible), finally we obtain the following upper triangular system which is equivalent to the original system of equations:

$$\begin{array}{ccccccc}
a_{11}^{(1)} x_1 & + & a_{12}^{(1)} x_2 & + \cdots + & a_{1m}^{(1)} x_m & = & b_1^{(1)} \\
 & & a_{22}^{(2)} x_2 & + \cdots + & a_{2m}^{(2)} x_m & = & b_2^{(2)} \\
 & & & \ddots & \vdots & & \vdots \\
 & & & & a_{mm}^{(m)} x_m & = & b_m^{(m)}
\end{array}$$

which can be solved by back-substitution. Note that in this form each row has a different superscript. The superscript represents the 'age' of the coefficients in that row, which is equal to the number of times the row has undergone ERO-2. It is more likely that values which were subject to many transformations are less accurate than others with fewer transformations.

If the matrix turned out to be singular, at the end of the above procedure we either get a contradiction (no solution exists) or obtain an equation in several variables. One of them can be expressed in term of the others. The 'others' can be given arbitrary values (infinitely many possibilities) and they determine all other variables via back-substitution.

Summarizing the above, step k of the triangularization phase of the Gaussian elimination is:

TrStep(k): Elimination of x_k from equations $k+1, \ldots, m$ (coefficients with superscript $^{(k)}$ are involved):

k.1 Look for a nonzero coefficient (in column of x_k) $a_{jk}^{(k)} \neq 0$, $j \geq k$, and swap row j with row k, i.e., perform `swap(row(j),row(k))`. This results in $a_{kk}^{(k)} \neq 0$.

k.2 For $j = k+1, \ldots, m$ perform ERO-2 on row j with row k using multiplier $a_{jk}^{(k)}/a_{kk}^{(k)}$.

Triangularization is achieved by performing TrStep(k) for $k = 1, \ldots, m-1$. It is then followed by back-substitution which completes the solution of (28) by Gaussian elimination.

There are several subtleties of this procedure. The most important of them is that in step k.1 it is worth looking for a j for which $|a_{jk}^{(k)}|$ is the largest and not simply nonzero. This choice can dramatically improve the numerical stability of the procedure.

There is an interesting modification of the above procedure which is referred to as *Gauss-Jordan elimination.* This is based on the observation that the back-substitution of Gaussian elimination may be avoided by adding row operations that make entries above the diagonal zero. The operations (32) and (33) are replaced by the operations

$$
\begin{aligned}
a_{ij}^{(k+1)} &= a_{ij}^{(k)} - \left(a_{ik}^{(k)}/a_{kk}^{(k)}\right) a_{kj}^{(k)}, \quad i \neq k,\ j > k \\
b_{i}^{(k+1)} &= b_{i}^{(k)} - \left(a_{ik}^{(k)}/a_{kk}^{(k)}\right) b_{k}^{(k)}, \quad i \neq k,
\end{aligned}
$$

and

$$b_{k}^{(k+1)} = b_{k}^{(k)},$$

and the system is reduced to the diagonal form

$$a_{kk}^{(k)} x_k = b_k^{(m)} \quad \text{for } k = 1, 2, \ldots, m,$$

which is trivial to solve. Note that here the superscript of b_k is $^{(m)}$ since the entire right-hand-side is transformed in each step. A slight modification of the G-J procedure is to apply ERO-1 to row k with $a_{kk}^{(k)}$ to make the coefficient of x_k equal to 1. In this case the final stage is

$$x_k = b_k^{(m)} \quad \text{for } k = 1, 2, \ldots, m,$$

which is the solution itself. This latter 'normalized' form is computationally not attractive because of some extra work but it has significance in linear programming.

Example 34 *G-J elimination:*

$$
\begin{array}{rcrcrcrl}
 & & & & & & & \textit{Operations to perform} \\
2x_1 & - & 3x_2 & + & 2x_3 & = & -2 & \\
x_1 & + & \frac{1}{2}x_2 & - & x_3 & = & 1 & \texttt{row(2):= row(2) - } \frac{1}{2} \texttt{ row(1)} \\
4x_1 & - & x_2 & + & x_3 & = & 3 & \texttt{row(3):= row(3) - } \frac{4}{2} \texttt{ row(1)} \\
\hline
2x_1 & - & 3x_2 & + & 2x_3 & = & -2 & \texttt{row(1):= row(1) - } \frac{(-3)}{2} \texttt{ row(2)} \\
 & & 2x_2 & - & 2x_3 & = & 2 & \\
 & & 5x_2 & - & 3x_3 & = & 7 & \texttt{row(3):= row(3) - } \frac{5}{2} \texttt{ row(2)} \\
\hline
2x_1 & & & - & x_3 & = & 1 & \texttt{row(1):= row(1) - } \frac{(-1)}{2} \texttt{ row(3)} \\
 & & 2x_2 & - & 2x_3 & = & 2 & \texttt{row(2):= row(2) - } \frac{(-2)}{2} \texttt{ row(3)} \\
 & & & & 2x_3 & = & 2 & \\
\hline
2x_1 & & & & & = & 2 & \\
 & & 2x_2 & & & = & 4 & \\
 & & & & 2x_3 & = & 2 &
\end{array}
$$

from which the solution is $x_1 = 1,\ x_2 = 2,\ x_3 = 1$.

3.5.3 Symmetric systems, Cholesky factorization

When matrix $\mathbf{A}$ is symmetric ($\mathbf{A} = \mathbf{A}^T$) and the G-J elimination steps can be performed without row changes then the symmetry of the transformed matrix is retained throughout the process. It means whenever the lower triangularization eliminates a column from the remaining matrix, the same automatically takes place with the corresponding row. (In this way, half of the computational work can be saved.)

If $\mathbf{A}$ is not only symmetric but also positive definite ($\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for any $\mathbf{x} \neq \mathbf{0}$) then some interesting properties ensure that the diagonal element can always be used for pivoting. These properties are:

1. All diagonal elements are positive ($a_{ii} > 0$) for all i.
2. The largest element (in magnitude) in the entire matrix occurs in the diagonal.
3. All the eigenvalues of $\mathbf{A}$ are strictly positive.
4. All leading principle minors (i.e., the $1 \times 1,\ 2 \times 2, \dots, m \times m$ submatrices beginning in the upper left corner) are positive definite.
5. If the Gaussian elimination without row changes is performed on $\mathbf{A}$, the remaining matrix is positive definite at every step.

In this case, the $\mathbf{A} = \mathbf{LU}$ factored form is also symmetric with $\mathbf{U} = \mathbf{L}^T$, i.e.,

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

This is called the *Cholesky factorization* of $\mathbf{A}$ and $\mathbf{L}$ is called the Cholesky factor.

The Cholesky factor can be computed in several different ways. One of them is the Gaussian elimination (not discussed). Another possibility is to determine its elements by direct computation. Writing out the elements of $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, we have:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix} = \begin{bmatrix} \ell_{11} & & & \\ \ell_{21} & \ell_{22} & & \\ \vdots & \vdots & \ddots & \\ \ell_{m1} & \ell_{m2} & \dots & \ell_{mm} \end{bmatrix} \begin{bmatrix} \ell_{11} & \ell_{21} & \dots & \ell_{m1} \\ & \ell_{22} & \dots & \ell_{m2} \\ & & \ddots & \vdots \\ & & & \ell_{mm} \end{bmatrix}$$

Here, a_{11} is defined by the dot product of the first row of $\mathbf{L}$ by the first column of $\mathbf{L}^T$ (which is equal to the first row of $\mathbf{L}$), giving $a_{11} = \ell_{11}^2$. Since property 1 ensures that $a_{11} > 0$, the (1,1) element of $\mathbf{L}$ is defined by $\ell_{11} = \sqrt{a_{11}}$ (> 0).

Continuing the dot products with row 1 of $\mathbf{L}$ and columns $2, \dots, m$ of $\mathbf{L}^T$ ($\equiv$ rows $2, \dots, m$ of $\mathbf{L}$), we obtain $\ell_{11}\ell_{21} = a_{12}, \dots, \ell_{11}\ell_{j1} = a_{1j}, \dots, \ell_{11}\ell_{m1} = a_{1m}$, which gives

$$\ell_{j1} = \frac{a_{1j}}{\ell_{11}}, \quad j = 2, \dots, m$$

completing the first column of $\mathbf{L}$.

Element (2,2) of $\mathbf{A}$ is defined as the dot product of row 2 of $\mathbf{L}$ and column 2 of $\mathbf{L}^T$:

$$\ell_{21}^2 + \ell_{22}^2 = a_{22} \quad \Rightarrow \quad \ell_{22}^2 = a_{22} - \ell_{21}^2.$$

Since $a_{22} - \ell_{21}^2$ is the leading diagonal element of the remaining matrix after one step of the Gaussian elimination, it is positive by property 5. Therefore ℓ_{22} is well defined. The (2,j) element of $\mathbf{A}$ is obtained as the dot product of row 2 of $\mathbf{L}$ and column j of $\mathbf{L}^T$: $\ell_{21}\ell_{j1} + \ell_{22}\ell_{j2} = a_{2j}$ for $j = 3, \ldots, m$. Since a_{2j}, ℓ_{21}, and ℓ_{j1} are known, we have

$$\ell_{j2} = \frac{a_{2j} - \ell_{21}\ell_{j1}}{\ell_{22}}, \quad j = 3, \ldots, m$$

which determines column 2 of $\mathbf{L}$. Continuing in this fashion, at the beginning of step k columns 1 through $k-1$ of $\mathbf{L}$ have been computed and the k-th diagonal element of $\mathbf{L}$ is defined by the equation

$$\ell_{kk}^2 = a_{kk} - \ell_{k1}^2 - \ell_{k2}^2 - \cdots - \ell_{k,k-1}^2. \tag{34}$$

Again, property 5 ensures that this quantity remains positive for $k = 1, \ldots, m$. The k-th column of $\mathbf{L}$ is determined from the relation

$$\ell_{k1}\ell_{j1} + \ell_{k2}\ell_{j2} + \cdots + \ell_{kk}\ell_{jk} = a_{kj}, \quad j = k+1, \ldots, m.$$

Below, we give a computational description of the Cholesky factorization. First, two functions, $cdiv(k)$ and $cmod(j, k)$, are defined, then the algorithm itself. It is assumed that m is globally available for the functions.

$cdiv(k)$	$cmod(j, k)$

$\ell_{kk} = \sqrt{a_{kk}}$
do $i = k+1$ to m
   $\ell_{ik} := a_{ik}/\ell_{kk}$
enddo
	do $i = j$ to m
	   $a_{ij} := a_{ij} - \ell_{ik}\ell_{jk}$
	enddo

Column Cholesky algorithm

```
do $k = 1$ to $m$
   $cdiv(k)$
   do $j = k+1$ to $m$
      $cmod(j, k)$
   enddo
enddo
```

The Cholesky factorization is an all-important computational technique in many different areas ranging from linear algebra through least squares problems to linear and nonlinear optimization. It can be used to determine whether a symmetric matrix $\mathbf{A}$ is positive definite or not. If the procedure terminates successfully then $\mathbf{A}$ is positive definite. If it breaks down at an intermediate stage because the right hand side of (34) becomes negative or zero, then $\mathbf{A}$ is not positive definite.

Example 35 *Find the Cholesky factor for*

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 10 & -1 \\ 1 & -1 & 5 \end{bmatrix}$$

It is symmetric but we do not know (yet) if it is positive definite.

$$\begin{bmatrix} \ell_{11} & & \\ \ell_{21} & \ell_{22} & \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \begin{bmatrix} \ell_{11} & \ell_{21} & \ell_{31} \\ & \ell_{22} & \ell_{32} \\ & & \ell_{33} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 10 & -1 \\ 1 & -1 & 5 \end{bmatrix}$$

$k = 1:$

$$\ell_{11}^2 = 1 \;\Rightarrow\; \ell_{11} = 1$$
$$\ell_{11}\ell_{21} = -1 \;\Rightarrow\; \ell_{21} = -1$$
$$\ell_{11}\ell_{31} = 1 \;\Rightarrow\; \ell_{31} = 1$$

$k = 2:$

$$\ell_{21}^2 + \ell_{22}^2 = 10 \;\Rightarrow\; \ell_{22}^2 = 9 \;\Rightarrow\; \ell_{22} = 3$$
$$\ell_{21}\ell_{31} + \ell_{22}\ell_{32} = -1 \;\Rightarrow\; \ell_{32} = 0$$

$k = 3:$

$$\ell_{31}^2 + \ell_{32}^2 + \ell_{33}^2 = 5 \;\Rightarrow\; \ell_{33}^2 = 4 \;\Rightarrow\; \ell_{33} = 2$$

Thus the factorization was successful $\Rightarrow$ **A** *is positive definite. Its factored form is*

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 10 & -1 \\ 1 & -1 & 5 \end{bmatrix} = \begin{bmatrix} 1 & & \\ -1 & 3 & \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \\ & 3 & 0 \\ & & 2 \end{bmatrix}$$

If $\mathbf{A}$ is in the Cholesky factored form, $\mathbf{Ax} = \mathbf{b}$ can be solved in the following way. Since now $\mathbf{A} = \mathbf{LL}^T$ (a special form of the LU decomposition), we have to solve $\mathbf{LL}^T\mathbf{x} = \mathbf{b}$. Denoting $\mathbf{L}^T\mathbf{x} = \mathbf{y}$, first $\mathbf{Ly} = \mathbf{b}$ is solved for $\mathbf{y}$ by forward substitution then $\mathbf{L}^T\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$ by back-substitution.

3.5.4 Gauss-Jordan elimination for the $m < n$ case

So far we have discussed the Gaussian (and Gauss-Jordan, G-J) elimination to solve $\mathbf{Ax} = \mathbf{b}$ when $\mathbf{A}$ is a square matrix. We know solution exists if $\mathbf{b} \in \text{range}(\mathbf{A})$. This latter is also true if $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m < n$ (fewer equations than variables). Obviously, in this case the column vectors of $\mathbf{A}$ are linearly dependent.

This subsection investigates the solution of $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m < n$. The normalized version of the G-J elimination can be used to make the coefficients of as many $x_k^{(k)} = 1$ as possible. The transformations are performed on the augmented matrix $[\mathbf{A} \mid \mathbf{b}]$. Its current transformed version is now denoted by $[\bar{\mathbf{A}} \mid \bar{\mathbf{b}}]$.

First, row 1 is searched for a pivot element. Assume, $a_{11} \neq 0$. To make it equal to 1 (as required by the normalized G-J) row 1 is divided by a_{11} and then x_1 is eliminated from all other rows by appropriate ERO-2s. Next, row 2 is taken. It has a zero in position 1 as the result of the previous elimination step. If $\bar{a}_{22} \neq 0$ variable x_2 can be eliminated from all rows, including row 1, in a similar fashion as in step 1. Assume this procedure can be repeated for all subsequent rows until step k ($\bar{a}_{ii} \neq 0,\ i = 1, \ldots, k-1$).

If $\bar{a}_{kk} = 0$ (and, of course, $\bar{a}_{ki} = 0,\ i = 1, \ldots, k-1$), it cannot be selected as pivot. In this case search continues in row k to locate a nonzero entry that can be used as a pivot element. If we find $\bar{a}_{kj} \neq 0$ then variable x_j can be eliminated from all other rows. The situation is visually represented in the following table.

	x_1				x_{k-1}	x_k		x_j		$\bar{\mathbf{b}}$
1	1	0	$\dots$	0	0	$\bar{a}_{1k}$		0		$\bar{b}_1$
	0	1								
	$\vdots$		$\ddots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$
	0	0		1	0					
$k-1$	0	0	$\dots$	0	1			0		$\bar{b}_{k-1}$
k	0	0	$\dots$	0	0	$\boxed{\mathbf{0}}$		$\boxed{1}$		$\bar{b}_k$
m	0	0	$\dots$	0	0	$\bar{a}'_{mk}$		0		$\bar{b}_m$

It may also happen that all $\bar{a}_{kj}$ entries are zero. In this situation two cases can be distinguished.

1. If $\bar{b}_k = 0$, row i is linearly dependent. It means the rank of $\mathbf{A}$ is less than m (rank deficiency) but the algorithm can proceed with the next row. We also say that such a row is *redundant*.
2. If $\bar{b}_k \neq 0$, system is incompatible: $\mathbf{b} \notin \text{range}(\mathbf{A})$, therefore, solution does not exist.

If there are redundant rows they simply can be dropped from the system of equations. We now assume $\mathbf{A}$ is of full (row) rank. It entails the above procedure finds m variables that are eliminated from all but one equations. The corresponding columns are unit vectors in the final matrix. Variables with a unit column vector are called *basic variables (BVs)* . All other variables are called *nonbasic variables (NBVs)*. Basic variables are associated with the rows of the matrix. For instance, in the above tabular form the basic variable of row $k-1$ is x_{k-1} and the BV of row k is x_j.

At the successful termination of the G-J procedure we have m basic and $n-m$ nonbasic variables. If the columns of the basic variables are permuted to the first m positions we have the following structure:

BV			NBV	$\bar{\mathbf{b}}$
1				
	$\ddots$			
		1		

Formally, the i-th equation is

$$x_i + \sum_{j=m+1}^{n} \bar{a}_{ij} x_j = \bar{b}_i,$$

from which the i-th basic variable can be expressed as

$$x_i = \bar{b}_i - \sum_{j=m+1}^{n} \bar{a}_{ij} x_j,$$

We can give arbitrary values to NBVs. Each such assignment *uniquely determines* the values of the BVs. Therefore, we have infinitely many solutions. For example, one possibility is to assign all zeros to the NBVs which yields $x_i = \bar{b}_i$. The freedom in assigning arbitrary values to NBVs can be utilized to generate solutions that satisfy some additional requirements, like all components are nonnegative or integer. These ideas are further discussed in the discipline of linear programming. A complete example of the G-J solution procedure is the subject of a tutorial.

In general, there are several possible choices of basic variables. For computational stability, it is advantageous to choose in each row an entry with the largest magnitude. In manual computing we usually look for a unit element in a row because in this case the normalization step (dividing the row by the pivot element) can be avoided.

The main point of this subsection is that the Gauss-Jordan elimination can be used to solve $\mathbf{Ax} = \mathbf{b}$ even if there are more variables than equations, $m < n$.

3.6 Solution of Systems of Linear Differential Equations

The solution of linear differential equations is an important application of computational linear algebra. This section is an introduction to this topic.

As a reminder, let us consider a single differential equation with an initial condition in the following form:

$$x'(t) = \lambda x(t), \quad x(0) = x^*, \tag{35}$$

where $x(t)$ is an unknown function of t, prime denotes differentiation and λ is a constant. It is known that the solution of (35) is

$$x(t) = x^* e^{\lambda t}. \tag{36}$$

Obviously, differentiating (36), we obtain (35).

There may be functional relationships among several functions $x_1(t), \ldots, x_m(t)$ and their derivatives. As an example, let us investigate the following system of linear differential equations with constant coefficients:

$$\begin{aligned} x_1' &= x_1 \\ x_2' &= x_1 + 2x_2 \\ x_3' &= x_1 - x_3. \end{aligned} \tag{37}$$

For notational simplicity, here we wrote x_i instead of $x_i(t)$, for $i = 1, 2, 3$. Introducing matrix notation, (37) can be expressed as

$$\mathbf{x}' = \mathbf{Ax}, \quad \text{where } \mathbf{x}' = \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{and } \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & -1 \end{bmatrix} \tag{38}$$

(38) can be investigated with initial conditions in the following general form:

$$\mathbf{x}' = \mathbf{Ax}, \quad \mathbf{x}(0) = \mathbf{x}^*. \tag{39}$$

We use the following theorem (presented without proof):

Theorem 1 *If* $\mathbf{A}$ *is a nonsingular* $m \times m$ *matrix that has distinct real eigenvalues then there exists a nonsingular matrix* $\mathbf{R}$ *such that* $\mathbf{RAR}^{-1} = \mathbf{D} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_m\}$, *where* $\lambda_1, \ldots, \lambda_m$ *are the eigenvalues of* $\mathbf{A}$.

Introducing new coordinates $\mathbf{y} = \mathbf{Rx}$ (from which $\mathbf{x} = \mathbf{R}^{-1}\mathbf{y}$), we can write

$$\mathbf{y}' = \mathbf{Rx}' = \mathbf{RAx} = \mathbf{RA}(\mathbf{R}^{-1}\mathbf{y}) = \mathbf{RAR}^{-1}\mathbf{y} = \mathbf{Dy},$$

or, in short,

$$\mathbf{y}' = \mathbf{Dy}. \tag{40}$$

Since $\mathbf{D}$ is diagonal, (40) decomposes into independent differential equations:

$$y_i' = \lambda_i y_i, \quad i = 1, \ldots, m. \tag{41}$$

We know that (41) has unique solution for every initial condition $y_i(0)$, namely:

$$y_i(t) = y_i(0)e^{\lambda_i t}, \quad i = 1, \ldots, m. \tag{42}$$

To solve (39), set $\mathbf{y}(0) = \mathbf{R}\mathbf{x}^*$. If $\mathbf{y}(t)$ denotes the solution of (40) then the solution of (39) is $\mathbf{x}(t) = \mathbf{R}^{-1}\mathbf{y}(t)$, or

$$\begin{bmatrix} x_1(t) \\ \vdots \\ x_m(t) \end{bmatrix} = \mathbf{R}^{-1} \begin{bmatrix} y_1(0)e^{\lambda_1 t} \\ \vdots \\ y_m(0)e^{\lambda_m t} \end{bmatrix}. \tag{43}$$

Differentiation shows that it is a solution to (39), because

$$\mathbf{x}' = \mathbf{R}^{-1}\mathbf{y}' = \mathbf{R}^{-1}\mathbf{D}\mathbf{y} = \mathbf{R}^{-1}(\mathbf{R}\mathbf{A}\mathbf{R}^{-1})\mathbf{y} = \mathbf{A}\mathbf{R}^{-1}\mathbf{y} = \mathbf{A}\mathbf{x}.$$

Comparing the beginning and the end of the equation, we have the required $\mathbf{x}' = \mathbf{A}\mathbf{x}$. For the initial conditions:

$$\mathbf{x}(0) = \mathbf{R}^{-1}\mathbf{y}(0) = \mathbf{R}^{-1}\mathbf{R}\mathbf{x}^* = \mathbf{x}^*,$$

which proves that $\mathbf{x}$ of (43) really solves (39). Furthermore, the solution is unique. The reason for it is that an $\mathbf{x}(t)$ is a solution to (39) if and only if $\mathbf{R}\mathbf{x}(t)$ is a solution to

$$\mathbf{y}' = \mathbf{D}\mathbf{y}, \quad \mathbf{y}(0) = \mathbf{R}\mathbf{x}^*. \tag{44}$$

Therefore, if there were two different solutions they would lead to two different solutions to (44), which is impossible since $\mathbf{D}$ is diagonal.

To solve the problem computationally, we need to determine $\mathbf{R}$. It can be shown that $\mathbf{R}$ is the inverse of the matrix formed by the eigenvectors of $\mathbf{A}$. Since, by assumption, the eigenvalues of $\mathbf{A}$ are distinct (and different from zero) the eigenvectors are also distinct. Additionally, they are linearly independent, thus $\mathbf{R}$ is nonsingular.

The first step is to compute the eigenvalues of $\mathbf{A}$. They can be obtained as the roots of the m-degree polynomial $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. By assumption, these roots are distinct now. Next, the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$ have to be determined by solving m sets of linear equations:

$$(\mathbf{A} - \lambda_i\mathbf{I})\mathbf{u}_i = \mathbf{0} \quad \text{for } i = 1, \ldots, m. \tag{45}$$

The matrix formed by the eigenvectors is denoted by $\mathbf{U}$:

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m] = \begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1m} \\ u_{21} & u_{22} & \ldots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \ldots & u_{mm} \end{bmatrix}.$$

Now, $\mathbf{R} = \mathbf{U}^{-1}$ and $\mathbf{R}^{-1} = \mathbf{U}$. Applying these identities to (43), we obtain

$$x_i(t) = \sum_{j=1}^{m} u_{ij} y_j(t) \quad \text{for } i = 1, \ldots, m, \tag{46}$$

where $y_j(t) = y_j(0)e^{\lambda_j t}$. To find a solution $\mathbf{x}(t)$ with a specified initial value

$$\mathbf{x}(0) = \mathbf{x}^* = [x_1^*, \ldots, x_m^*]^T,$$

we substitute $t = 0$ into (46) and solve the system of linear equations for unknowns $z_1 = y_1(0), \ldots, z_m = y_m(0)$:

$$\sum_{j=1}^{m} u_{ij} z_j = x_i^* \quad \text{for } i = 1, \ldots, m,$$

or in matrix notation, $\mathbf{U}\mathbf{z} = \mathbf{x}^*$, from which

$$\mathbf{z} = \mathbf{U}^{-1}\mathbf{x}^*.$$

In other words, the initial values $\mathbf{x}(0) = \mathbf{x}^*$ correspond to the initial values $\mathbf{y}(0) = \mathbf{U}^{-1}\mathbf{x}^*$ of (42).

Example 36 *Find a general solution to the introductory example in (37)*

$$\begin{aligned} x_1' &= x_1 \\ x_2' &= x_1 + 2x_2 \\ x_3' &= x_1 - x_3 \end{aligned}$$

which is in matrix form

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{or} \quad \mathbf{x}' = \mathbf{A}\mathbf{x}$$

Since $\mathbf{A}$ *is triangular,* $\det(\mathbf{A} - \lambda\mathbf{I}) = (1-\lambda)(2-\lambda)(-1-\lambda)$, *from which the eigenvalues are* $\lambda_1 = 1$, $\lambda_2 = 2$, *and* $\lambda_3 = -1$. *They are real and distinct so the assumptions of the theorem are satisfied. Therefore, the diagonal matrix* $\mathbf{D}$ *is*

$$\mathbf{D} = \begin{bmatrix} 1 & & \\ & 2 & \\ & & -1 \end{bmatrix}$$

and the new (equivalent) set of differential equations is

$$\begin{aligned} y_1' &= y_1 \\ y_2' &= 2y_2 \\ y_3' &= -y_3, \end{aligned}$$

which has the solution

$$\begin{aligned} y_1(t) &= c_1 e^{t} \\ y_2(t) &= c_2 e^{2t} \\ y_3(t) &= c_3 e^{-t}, \end{aligned}$$

where c_1, c_2, c_3 *are arbitrary constants. The eigenvectors corresponding to the eigenvalues can be determined by (45). In our case, however, by inspection we can identify* $\mathbf{u}_2$ *as* $[0,1,0]^T$ *and* $\mathbf{u}_3$ *as* $[0,0,1]^T$. *To find* $\mathbf{u}_1$ *we have to solve* $(\mathbf{A} - \mathbf{I})\mathbf{u}_1 = \mathbf{0}$, *which is*

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & -2 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{21} \\ u_{31} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

A nontrivial solution to this system is $\mathbf{u}_1 = [2, -2, 1]^T$. *Now we can form matrix* $\mathbf{U}$ *which is:*

$$\mathbf{U} = \begin{bmatrix} 2 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Applying (43), we obtain

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 e^t \\ c_2 e^{2t} \\ c_3 e^{-t} \end{bmatrix} \tag{47}$$

which gives the general solution as

$$\begin{aligned} x_1(t) &= 2c_1 e^t \\ x_2(t) &= -2c_1 e^t + c_2 e^{2t} \\ x_3(t) &= c_1 e^t + c_3 e^{-t} \end{aligned}$$

with arbitrary constants c_1, c_2, c_3.

If an initial value problem is to be solved with some $x_i(0) = x_i^*,\ i = 1, 2, 3,$ then the values for c_1, c_2 and c_3 have to be determined appropriately by substituting $t = 0$ in (47)

$$\begin{bmatrix} 2 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \end{bmatrix},$$

which is simply a system of linear equations that can be solved by using any suitable method.

3.7 Least Squares Problems

One of the most important application areas of computational linear algebra is least squares problems. Such problems occur in many diverse situations. The prototype problem is that some constants (called *parameters*) of a functional relationship have to be determined so that the resulting function matches (approximates) a set of observed data in the best possible way. The quality of the fit is measured by a norm of the difference between the vector of the observed and computed data. The norm is usually the Euclidean (ℓ_2) norm because it has some favourable properties.

The forthcoming discussion will be restricted to the most important case: the *linear least squares problems*, where the functional relationship is linear in the parameters. For example, if $a,\ b$ and c are three parameters then $y = ax^2 + bx + c$ is linear in $a,\ b$ and c, while $y = x^a + bx + c$ is not. The nonlinear least squares problems are beyond the scope of this course.

3.7.1 Formulation of the least squares problem

Having studied the general $\mathbf{Ax} = \mathbf{b}$ system of linear equations, we know that there are cases when this system is unsolvable (incompatible). For instance, if $\mathbf{A} \in \mathbb{R}^{3\times 1}$ (a matrix with 3 rows and 1 column) and $\mathbf{b} \in \mathbb{R}^3$ with the following coefficients:

$$\mathbf{A} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

then there is no multiple of the only column of $\mathbf{A}$ that is equal to $\mathbf{b}$. However, even in this case we might be interested in finding an $\mathbf{x}$ for which $\mathbf{Ax}$ approximates $\mathbf{b}$ the best. In the general case of an $m \times n$ incompatible $\mathbf{Ax} = \mathbf{b}$ system, instead of satisfying $\mathbf{Ax} = \mathbf{b}$, we want to find an $\mathbf{x}$ such that $\mathbf{Ax}$ is as close to $\mathbf{b}$ as possible, i.e., an $\mathbf{x}$ that minimizes $\|\mathbf{b} - \mathbf{Ax}\|_2$. In practice, it is more convenient to use the square of the norm and define the least squares problem as to

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\ \|\mathbf{b} - \mathbf{Ax}\|_2^2. \tag{48}$$

The actual difference between $\mathbf{b}$ and $\mathbf{Ax}$ is called *residual* and is denoted by $\boldsymbol{\rho}$:

$$\boldsymbol{\rho} = \mathbf{b} - \mathbf{Ax}. \tag{49}$$

Using this terminology, the (48) least squares problem minimizes the square of the Euclidean norm of the residual (49). For any given $\mathbf{x}$, the residual of row i is $\rho_i = b_i - \mathbf{a}^i\mathbf{x}$, where $\mathbf{a}^i\mathbf{x}$ is the dot product of the i-th row of matrix $\mathbf{A}$ with $\mathbf{x}$.

From previous discussion we know that if the system is compatible then there exists a linear combination $\mathbf{Ax}$ of the columns of $\mathbf{A}$ that gives $\mathbf{b}$ exactly. In such a case the residual is $\mathbf{0}$.

In the sequel, we will refer to the data fitting model, where n parameters $x_1, \dots, x_n$ of a linear relationship have to be determined from m observations. (Remember the introductory model: amount of heating oil as a function of the outside temperature and amount of attic insulation.) In the i-th observation we have the following figures: b_i (the value of the dependent variable to be 'explained') and $a_{i1}, \dots, a_{in}$ (the observed values of the 'explaining' variables). The weighted sum of the observed data should approximate the observed value of the dependent variable:

$$b_i \approx a_{i1}x_1 + \cdots + a_{in}x_n = \sum_{j=1}^{n} a_{ij}x_j, \; i = 1, \dots, m.$$

The solution $\mathbf{x}$ of a linear least squares problem is an n-vector that produces the smallest sum of squared errors in the model over a given set of observations, i.e., that solves

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij}x_j \right)^2 .$$

If

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \; \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \; \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

then the model can be represented in matrix form as $\min \|\mathbf{b} - \mathbf{Ax}\|_2^2$.

If $m > n$, the least squares problem is called *overdetermined* because a linear model with n free coefficients cannot in general exactly match a number of observations larger than n. In practical cases it is nearly always true that $m > n$, sometimes even $m \gg n$.

3.7.2 Solution of the least squares problem

Subsection 3.3.2 introduced the notion of subspaces defined by the rows and columns of an $m \times n$ matrix $\mathbf{A}$. It is useful to think of a least squares problem involving matrix $\mathbf{A}$ in terms of the subspaces defined by $\mathbf{A}$. The range space of $\mathbf{A}$, range($\mathbf{A}$), consists of all m-vectors that are linear combinations of the columns of $\mathbf{A}$. The complementary subspace, null($\mathbf{A}^T$), consists of all m-vectors $\mathbf{z}$ orthogonal to the columns of $\mathbf{A}$, i.e., $\mathbf{a}_j^T\mathbf{z} = 0$ for $j = 1, \dots, n$, (or, with matrix notation, $\mathbf{A}^T\mathbf{z} = \mathbf{0}$) if $\mathbf{z} \in$ null($\mathbf{A}^T$). Let r denote the rank of $\mathbf{A}$ (assume $r > 0$). The dimension of range($\mathbf{A}$) is r, and the dimension of null($\mathbf{A}^T$) is $m - r$.

We know that, given $\mathbf{A}$, any m-vector $\mathbf{b}$ can be written uniquely as the sum of its range and null space components, i.e., $\mathbf{b} = \mathbf{b}_R + \mathbf{b}_N$, where $\mathbf{b}_R \in$ range($\mathbf{A}$) and $\mathbf{b}_N \in$ null($\mathbf{A}^T$), i.e., $\mathbf{b}_R^T\mathbf{b}_N = 0$. Similar arguments apply to the residual: $\boldsymbol{\rho} = \boldsymbol{\rho}_R + \boldsymbol{\rho}_N$ and $\boldsymbol{\rho}_R^T\boldsymbol{\rho}_N = 0$. Using these notations,

$$\boldsymbol{\rho} = \boldsymbol{\rho}_R + \boldsymbol{\rho}_N = \mathbf{b} - \mathbf{Ax} = \mathbf{b}_R + \mathbf{b}_N - \mathbf{Ax} = \underbrace{\mathbf{b}_R - \mathbf{Ax}} + \underbrace{\mathbf{b}_N} .$$

Since by definition $\mathbf{Ax} \in$ range($\mathbf{A}$), the range space component of $\mathbf{b} - \mathbf{Ax}$ is $\mathbf{b}_R - \mathbf{Ax}$ which entails that its null space component is $\mathbf{b}_N$, i.e., $\boldsymbol{\rho}_R = \mathbf{b}_R - \mathbf{Ax}$ and $\boldsymbol{\rho}_N = \mathbf{b}_N$. Therefore, we can conclude that

$$\begin{aligned}
\|\mathbf{b}-\mathbf{Ax}\|_2^2 &= \|\boldsymbol{\rho}\|_2^2 \\
&= \|\boldsymbol{\rho}_R + \boldsymbol{\rho}_N\|_2^2 \\
&= (\boldsymbol{\rho}_R + \boldsymbol{\rho}_N)^T(\boldsymbol{\rho}_R + \boldsymbol{\rho}_N) \\
&= \boldsymbol{\rho}_R^T\boldsymbol{\rho}_R + \underbrace{\boldsymbol{\rho}_R^T\boldsymbol{\rho}_N}_{=0} + \underbrace{\boldsymbol{\rho}_N^T\boldsymbol{\rho}_R}_{=0} + \boldsymbol{\rho}_N^T\boldsymbol{\rho}_N \\
&= \|\boldsymbol{\rho}_R\|_2^2 + \|\boldsymbol{\rho}_N\|_2^2 \\
&= \|\mathbf{b}_R - \mathbf{Ax}\|_2^2 + \|\mathbf{b}_N\|_2^2 \\
&\geq \|\mathbf{b}_N\|_2^2.
\end{aligned}$$

Comparing the left hand side and the penultimate row it is obvious that $\|\mathbf{b}-\mathbf{Ax}\|_2^2$ is minimized if $\|\mathbf{b}_R-\mathbf{Ax}\|_2^2$ is as small as possible. Additionally, the last row shows that the minimum cannot be smaller than $\|\mathbf{b}_N\|_2^2$ (which is constant for a given $\mathbf{A}$). However, this theoretical minimum can be achieved because, by definition, $\mathbf{b}_R \in \text{range}(\mathbf{A})$ and, therefore, there exists an $\mathbf{x}$ for which $\mathbf{Ax} = \mathbf{b}_R$. This $\mathbf{x}$ will make $\|\mathbf{b}_R-\mathbf{Ax}\|_2^2 = 0$. For this $\mathbf{x}$, the entire range space component of $\mathbf{b}$ is removed by subtraction of $\mathbf{Ax}$, and what remains is nothing but $\mathbf{b}_N$, i.e., $\mathbf{b}-\mathbf{Ax} = \mathbf{b}_N$.

Since $\mathbf{b}_N$ lies in the null space of $\mathbf{A}^T$, relation $\mathbf{A}^T\mathbf{b}_N = 0$ holds. If $\mathbf{b}-\mathbf{Ax}$ is substituted for $\mathbf{b}_N$, the following fundamental characterization of the optimal least squares solution is obtained:

$$\mathbf{x} \text{ minimizes } \|\mathbf{b}-\mathbf{Ax}\|_2^2 \text{ iff } \mathbf{A}^T(\mathbf{b}-\mathbf{Ax}) = \mathbf{0}. \tag{50}$$

This is equivalent to the following:

$$\mathbf{x} \text{ minimizes } \|\mathbf{b}-\mathbf{Ax}\|_2^2 \text{ iff } \mathbf{Ax} = \mathbf{b}_R \text{ and } \mathbf{b}-\mathbf{Ax} = \mathbf{b}_N. \tag{51}$$

(51) shows that if $\mathbf{x}$ is a minimizer then the residual is $\boldsymbol{\rho} = \mathbf{b}_N$, and (50) says that the residual is orthogonal to the columns of $\mathbf{A}$. This phenomenon is similar to the geometrical principle that the shortest distance from a point to a plane is the length of the perpendicular joining them. Now, the point is the vector $\mathbf{b}$, and the plane is all the vectors in the range of $\mathbf{A}$.

Since the range and null space components ($\mathbf{b}_R$ and $\mathbf{b}_N$) of $\mathbf{b}$ are unique, the optimal least squares residual vector is also unique (being $\mathbf{b}_N$ itself). What is not necessarily unique is the solution vector $\mathbf{x}$. However, it is true (proof omitted) that if the columns of $\mathbf{A}$ are linearly independent, even $\mathbf{x}$ is unique. Thus, we can conclude that *the solution of* $\min\limits_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{b}-\mathbf{Ax}\|_2^2$ *is unique iff* $\mathbf{A}$ *has full column rank.* In practical cases it is very likely that this condition is satisfied.

Having established that the optimal least squares residual lies in the null space of $\mathbf{A}^T$, see (50), we know that the solution $\mathbf{x}$ satisfies $\mathbf{A}^T(\mathbf{b}-\mathbf{Ax}) = \mathbf{0}$. Rearranging this equation, we obtain the *normal equations*:

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}. \tag{52}$$

The symmetric matrix $\mathbf{A}^T\mathbf{A}$ is positive semidefinite for any matrix $\mathbf{A}$, and positive definite iff the columns of $\mathbf{A}$ are linearly independent. The normal equations are always compatible. A solution of (52) exists even if $\mathbf{A}^T\mathbf{A}$ singular.

The normal equations are of great practical importance because they offer a straightforward way to compute the least squares solution. When $\mathbf{A}^T\mathbf{A}$ is positive definite, the normal equations have a unique solution. In this case we can use the Cholesky factorization to solve the linear least squares problem with the help of the following algorithm:

Step 1. Form the normal equations matrix $\mathbf{A}^T\mathbf{A}$ and the vector $\mathbf{A}^T\mathbf{b}$.

Step 2. Compute the Cholesky factorization $\mathbf{A}^T\mathbf{A} = \mathbf{LL}^T$.

Step 3. Solve the (52) normal equations by solving two triangular systems: $\mathbf{Ly} = \mathbf{A}^T\mathbf{b}$ for $\mathbf{y}$, and $\mathbf{L}^T\mathbf{x} = \mathbf{y}$ to obtain the desired solution $\mathbf{x}$.

Example 37 *Let*

$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 1 & 2 \\ 2 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 5 \\ -1 \end{bmatrix}.$$

The matrix of normal equations and the right hand side:

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 9 & 6 \\ 6 & 8 \end{bmatrix}, \quad \mathbf{A}^T\mathbf{b} = \begin{bmatrix} 3 \\ 10 \end{bmatrix}.$$

The normal equations:

$$\begin{array}{rcrcr} 9x_1 & + & 6x_2 & = & 3 \\ 6x_1 & + & 8x_2 & = & 10 \end{array}$$

Though this system can easily be solved with the Gaussian elimination, to demonstrate the use of the Cholesky factorization, we proceed as follows:

$$\mathbf{A}^T\mathbf{A} = \mathbf{L}\mathbf{L}^T = \begin{bmatrix} 3 & \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ & 2 \end{bmatrix}$$

The first triangular system is $\mathbf{Ly} = \mathbf{A}^T\mathbf{b}$*:*

$$\begin{array}{rcrcr} 3y_1 & & & = & 3 \\ 2y_1 & + & 2y_2 & = & 10 \end{array}$$

from which $y_1 = 1$ *and* $y_2 = 4$*. The next system is* $\mathbf{L}^T\mathbf{x} = \mathbf{y}$*:*

$$\begin{array}{rcrcr} 3x_1 & + & 2x_2 & = & 1 \\ & & 2x_2 & = & 4 \end{array}$$

which gives the solution $\mathbf{x} = [-1, 2]^T$*. Using this* $\mathbf{x}$*,* $\mathbf{b}_R = \mathbf{Ax}$ *gives* $\mathbf{b}_R = [2, 3, -2]^T$ *and* $\mathbf{b}_N = \mathbf{b} - \mathbf{Ax}$ *gives* $\boldsymbol{\rho} = \mathbf{b}_N = [-2, 2, 1]^T$*. The value of least squares is the Euclidean norm of* $\mathbf{b}_N$ *which is 3. This completes the solution of the least squares example.*

3.8 Condition of a Mathematical Problem

The *condition of a problem* is a measure that reflects the sensitivity of its *exact* solution to changes in the problem. To quantify this idea, assume that a problem P is defined by a set of data, say d. Let $s(d)$ denote the *exact solution* of the problem for the data d. If small changes in d lead to small changes in $s(d)$, the problem is said to be *well-conditioned* for the data d. Otherwise, if small changes in d lead to *large* changes in $s(d)$, the problem is said to be *ill-conditioned*. Suppose that d_1 and d_2 are two possible sets of data. The *condition* (or *condition number*) of problem P is a measure of the *maximum* possible ratio

$$\frac{\|s(d_1) - s(d_2)\|}{\|d_1 - d_2\|} \tag{53}$$

when $\|d_1 - d_2\|$ is small. The crucial point to remember is that the condition of a problem is a *mathematical* property, and is independent of computation and rounding error.

As a simple illustration of condition, consider the problem of determining the roots of the polynomial

$$(x - 1)^4 = 0, \tag{54}$$

whose four roots are all exactly equal to 1. Suppose that a small perturbation (say, 10^{-8}) is made in the right-hand side of (54), so that the equation to be solved is now

$$(x-1)^4 = 10^{-8}. \tag{55}$$

One *exact* root of (55) is $1 + 10^{-2}$, which has changed by 10^{-2} from the exact root of the original problem. By almost any standard, problem (55) is close to (54), since their right-hand sides differ by only 10^{-8}. However, the change in the exact solution is six orders of magnitude larger than the change in the problem! Since the ratio (53) is of order 10^6 (i.e., substantially larger than one), we say that problem (54) is *ill-conditioned*. Note that this property is not related to computation in any way.

3.8.1 Matrix norms revisited

For any vector norm, an associated matrix norm can be defined as follows. Intuitively, the matrix $\mathbf{A}$ should have a "large" norm if its application to a nonzero vector $\mathbf{x}$ can produce a vector $\mathbf{Ax}$ whose norm is large relative to $\|\mathbf{x}\|$. For any matrix $\mathbf{A}$ and any vector norm $\|\cdot\|$ the *subordinate* matrix norm $\|\mathbf{A}\|$ is defined as

$$\|\mathbf{A}\| = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}$$

Note that norms on the right hand side are vector norms. This is equivalent to

$$\|\mathbf{A}\| = \max_{\|\mathbf{u}\|=1} \|\mathbf{Au}\|$$

(why?). A lower bound on $\|\mathbf{A}\|$ can be obtained by computing the ratio of $\|\mathbf{Ax}\|$ and $\|\mathbf{x}\|$ for any specific nonzero vector $\mathbf{x}$:

$$\|\mathbf{A}\| \geq \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}.$$

3.8.2 Condition of a linear system with a nonsingular matrix

Let $\mathbf{A}$ be a nonsingular matrix, so that $\mathbf{A}^{-1}$ exists (and is unique), and consider the linear system

$$\mathbf{Ax} = \mathbf{b},$$

whose *exact* (unique) solution is $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Suppose that the right-hand side $\mathbf{b}$ of this system is perturbed to $\mathbf{b} + \delta\mathbf{b}$, while $\mathbf{A}$ remains unaltered. Let $\mathbf{x} + \delta\mathbf{x}_b$ denote the *exact* solution to the perturbed problem

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}_b) = \mathbf{b} + \delta\mathbf{b}. \tag{56}$$

In this section, the occurrence of "δ" immediately preceding the name of a vector or matrix means a perturbation of the same dimension, so that $\delta\mathbf{b}$ is a change in the vector $\mathbf{b}$. The subscript "b" on $\delta\mathbf{x}_b$ emphasizes that this change in the exact solution results from a change in $\mathbf{b}$.

Since $\mathbf{Ax} = \mathbf{b}$, relation (56) implies that $\mathbf{A}\delta\mathbf{x}_b = \delta\mathbf{b}$, and hence

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

Using any subordinate norm, we obtain a bound on $\|\delta\mathbf{x}_b\|$:

$$\|\delta\mathbf{x}_b\| \leq \|\mathbf{A}^{-1}\|\,\|\delta\mathbf{b}\|, \tag{57}$$

where equality will hold for at least one vector $\delta\mathbf{b}$. Relation (57) shows that the size of the change in the exact solution of $\mathbf{Ax} = \mathbf{b}$ resulting from a change $\delta\mathbf{b}$ in the right-hand side may be as large as $\|\mathbf{A}^{-1}\|$ *multiplied by* $\|\delta\mathbf{b}\|$.

To express (57) in terms of *relative perturbations*, note that the relation $\mathbf{Ax} = \mathbf{b}$ implies

$$\|\mathbf{b}\| \le \|\mathbf{A}\|\,\|\mathbf{x}\|, \tag{58}$$

where again equality is possible for certain $\mathbf{x}$ and $\mathbf{b}$. Multiplying (57) by (58) and rearranging, we obtain the following important result:

$$\frac{\|\delta\mathbf{x}_b\|}{\|\mathbf{x}\|} \le \|\mathbf{A}^{-1}\|\,\|\mathbf{A}\|\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}. \tag{59}$$

We emphasize that the quantity on the right-hand side is the *maximum possible value* of the relative change in the exact solution. Although the upper bound is unduly large in some cases, equality will hold in (59) for certain choices of $\mathbf{A}$, $\mathbf{b}$ and $\delta\mathbf{b}$.

Next, we perturb the matrix $\mathbf{A}$ while keeping $\mathbf{b}$ fixed, so that we are interested in the exact solution of

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}_A) = \mathbf{b}, \tag{60}$$

where the subscript "$_A$" signals an association with a change in $\mathbf{A}$. Assume that $\|\delta\mathbf{A}\|$ is small enough so that $\mathbf{A} + \delta\mathbf{A}$ remains nonsingular. A bound on the relative change in the exact solution can be obtained in the following way. Performing the multiplication in (60) we get

$$\mathbf{Ax} + \mathbf{A}\delta\mathbf{x}_A + \delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}_A) = \mathbf{b}.$$

Since $\mathbf{Ax} = \mathbf{b}$, this reduces to

$$\begin{aligned} \mathbf{A}\delta\mathbf{x}_A &= -\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}_A) \\ \delta\mathbf{x}_A &= -\mathbf{A}^{-1}\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}_A). \end{aligned}$$

For the norms:

$$\begin{aligned} \|\delta\mathbf{x}_A\| &\le \|\mathbf{A}^{-1}\|\,\|\delta\mathbf{A}\|\,\|\mathbf{x} + \delta\mathbf{x}_A\| \\ \frac{\|\delta\mathbf{x}_A\|}{\|\mathbf{x} + \delta\mathbf{x}_A\|} &\le \|\mathbf{A}^{-1}\|\,\|\delta\mathbf{A}\| \end{aligned}$$

Multiplying and dividing the right hand side by $\|\mathbf{A}\|$, we obtain the bound in the following form:

$$\frac{\|\delta\mathbf{x}_A\|}{\|\mathbf{x} + \delta\mathbf{x}_A\|} \le \|\mathbf{A}^{-1}\|\,\|\mathbf{A}\|\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}, \tag{61}$$

where equality holds for certain choices of $\delta\mathbf{A}$ and $\mathbf{b}$.

The quantity $\|\mathbf{A}^{-1}\|\,\|\mathbf{A}\|$ appears in both (59) and (61), and reflects the *maximum* possible relative change in the *exact* solution of a linear system induced by a change in the data (either the right-hand side or the matrix itself). Guided by (53), we define the *condition number* (or simply the *condition*) of a nonsingular matrix $\mathbf{A}$ (with respect to solving $\mathbf{Ax} = \mathbf{b}$) as

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}^{-1}\|\,\|\mathbf{A}\|. \tag{62}$$

The notation $\kappa(\mathbf{A})$ is sometimes used for $\mathrm{cond}(\mathbf{A})$.

For any subordinate norm, the identity matrix has norm one. Since $\mathbf{I} = \mathbf{A}^{-1}\mathbf{A}$ and $\|\mathbf{A}^{-1}\mathbf{A}\| \le \|\mathbf{A}^{-1}\|\,\|\mathbf{A}\|$, we conclude that $\mathrm{cond}(\mathbf{A}) \ge 1$. Therefore, a *well-conditioned* matrix has a condition number of order unity (and the identity matrix is "perfectly" conditioned). An *ill-conditioned* matrix has a condition number much larger than unity. Although the exact value of $\mathrm{cond}(\mathbf{A})$ will vary with the norm used in (62), its size will be comparable for any norm because of the relationship among norms.

According to (59) and (61), knowledge of the condition number allows us to bound the *maximum* relative change in the exact solution. Conversely, knowledge of the relative change in the exact solution for any *particular perturbation* in $\mathbf{b}$ or $\mathbf{A}$ allows us to obtain a *lower bound* on the condition number. Rearranging (59) and (61), we have

$$\operatorname{cond}(\mathbf{A}) \geq \frac{\|\delta\mathbf{x}_b\|/\|\mathbf{x}\|}{\|\delta\mathbf{b}\|/\|\mathbf{b}\|} \quad \text{and} \quad \operatorname{cond}(\mathbf{A}) \geq \frac{\|\delta\mathbf{x}_A\|/\|\mathbf{x}+\delta\mathbf{x}_A\|}{\|\delta\mathbf{A}\|/\|\mathbf{A}\|}. \tag{63}$$

These relations show that, if the relative change in the exact solution is much larger than the relative perturbation in $\mathbf{b}$ or $\mathbf{A}$, we know immediately that $\mathbf{A}$ is ill-conditioned.

Example 38 *Let us consider the following set of equations:*

$$\begin{array}{rcrcl} 0.550x_1 & + & 0.423x_2 & = & 0.127 \\ 0.484x_1 & + & 0.372x_2 & = & 0.112 \end{array}$$

which is in the form of $\mathbf{Ax} = \mathbf{b}$ *with*

$$\mathbf{A} = \begin{bmatrix} 0.550 & 0.423 \\ 0.484 & 0.372 \end{bmatrix} \quad \textit{and} \quad \mathbf{b} = \begin{bmatrix} 0.127 \\ 0.112 \end{bmatrix}.$$

We will see that this system is rather ill-conditioned. Consider the original vector $\mathbf{b} = [0.127, 0.112]^T$ *and the following perturbed vector*

$$\bar{\mathbf{b}} = \begin{bmatrix} 0.12707 \\ 0.11228 \end{bmatrix} = \begin{bmatrix} 0.127 \\ 0.112 \end{bmatrix} + \begin{bmatrix} 0.00007 \\ 0.00028 \end{bmatrix} = \mathbf{b} + \delta\mathbf{b}. \tag{64}$$

The exact solution of $\mathbf{Ax} = \mathbf{b}$ *is* $\mathbf{x} = [1, -1]^T$*, but the exact solution of* $\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ *is* $\bar{\mathbf{x}} = [1.7, -1.91]^T$*, so that* $\delta\mathbf{x}_b = [0.7, -0.91]^T$*. The relative perturbations in the right-hand side and solution, measured in the infinity-norm, are*

$$\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} = \frac{.00028}{.127} \approx 2.2 \times 10^{-3} \text{ and } \frac{\|\delta\mathbf{x}_b\|}{\|\mathbf{x}\|} = 0.91.$$

For this choice of $\delta\mathbf{b}$*, the relative change in the solution is more than 400 times the relative change in the right-hand side, which shows that the condition number of* $\mathbf{A}$ *is at least 400 (see (63)).*

The ill-conditioning of A *can also be observed by making a "small" perturbation (of 0.001) in the (2,1) element, producing*

$$\mathbf{A} + \delta\mathbf{A} = \begin{bmatrix} 0.550 & 0.423 \\ 0.483 & 0.372 \end{bmatrix}. \tag{65}$$

The exact solution, say $\bar{\mathbf{x}}$*, of* $(\mathbf{A} + \delta\mathbf{A})\bar{\mathbf{x}} = \mathbf{b}$ *is* $\bar{\mathbf{x}} = [-0.4536, 0.8900]^T$*, which gives* $\|\delta\mathbf{x}_A\| = \|\bar{\mathbf{x}} - \mathbf{x}\| = 1.89$*. In this case, the relative change in the exact solution is approximately 2000 times the relative change in* $\mathbf{A}$*.*

In fact, the exact inverse of $\mathbf{A}$ *(rounded to five digits) is*

$$\mathbf{A}^{-1} = \begin{bmatrix} -2818.2 & 3204.5 \\ 3666.7 & -4166.7 \end{bmatrix}.$$

Since $\|\mathbf{A}\| = 0.973$ *and* $\|\mathbf{A}^{-1}\| = 7833.4$*, the condition number of* $\mathbf{A}$ *(measured in the infinity-norm) is approximately 7622. Thus, neither* $\delta\mathbf{b}$ *of (64) nor* $\delta\mathbf{A}$ *of (65) produces the maximum possible change in the exact solution!*

A matrix $\mathbf{A}$ is ill-conditioned when its application to different vectors of the same size produces transformed vectors of dramatically different size. If $\|\mathbf{Ax}\|$ is large for one vector $\mathbf{x}$ of unit norm and small for another, then $\mathbf{A}$ is ill-conditioned. It is the *difference* in possible sizes of $\|\mathbf{Ax}\|$ that leads to ill-conditioning. If $\|\mathbf{Ax}\|/\|\mathbf{x}\|$ is large for *all* $\mathbf{x}$—for example, when $\mathbf{A} = \mathrm{diag}(10^8, 10^8)$—then $\mathbf{A}$ is not ill-conditioned. In contrast, consider the following matrix and vectors:

$$\bar{\mathbf{A}} = \begin{bmatrix} 10^4 & \\ & 10^{-4} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

for which $\|\mathbf{x}\| = \|\bar{\mathbf{x}}\| = 1$. We then have

$$\bar{\mathbf{A}}\mathbf{x} = \begin{bmatrix} 10^4 \\ 0 \end{bmatrix}, \text{ and } \bar{\mathbf{A}}\bar{\mathbf{x}} = \begin{bmatrix} 0 \\ 10^{-4} \end{bmatrix},$$

so that $\|\bar{\mathbf{A}}\mathbf{x}\| = 10^4$ (large) and $\|\bar{\mathbf{A}}\bar{\mathbf{x}}\| = 10^{-4}$ (small). Since vectors of unit norm transformed by $\bar{\mathbf{A}}$ vary enormously in size, $\bar{\mathbf{A}}$ must be ill-conditioned. In fact, it is easy to see that $\mathbf{x}$ and $\bar{\mathbf{x}}$ achieve the maximum and minimum possible changes in norm when transformed by $\bar{\mathbf{A}}$, and that $\mathrm{cond}(\bar{\mathbf{A}}) = 10^8$.

The condition of a nonsingular matrix $\mathbf{A}$ is a quantitative indication of the sensitivity to perturbation of a linear system involving $\mathbf{A}$. The condition of $\mathbf{A}$ also has a precise interpretation in terms of the closeness of $\mathbf{A}$ to singularity. Informally speaking, an ill-conditioned matrix is "nearly singular".

3.8.3 Condition of the normal equations

Solving the least squares problem by means of the normal equations is quite efficient in terms of computational work. The number of flops required to form $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ is of order $\frac{1}{2}mn^2$, computing the Cholesky factorization uses of order $\frac{1}{6}n^3$ flops, and solving the two triangular systems involves of order n^2 flops.

Unfortunately, use of the normal equations has undesirable implications with respect to numerical stability because the condition number of $\mathbf{A}^T\mathbf{A}$ is the square of the condition number of $\mathbf{A}$. Consequently, the normal equation matrix $\mathbf{A}^T\mathbf{A}$ can be severely ill-conditioned.

For example, let

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 10^{-4} \end{bmatrix}.$$

The condition of a nonsquare matrix was not discussed, therefore it is to be accepted that $\mathrm{cond}(\mathbf{A}) = 2.8 \times 10^4$ now (rounded to two figures). The associated normal equation matrix is

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 2 + 10^{-8} \end{bmatrix},$$

with $\mathrm{cond}(\mathbf{A}^T\mathbf{A}) = 7.8 \times 10^8$.

Ill-conditioning in the normal equations may lead not only to inaccuracy in the computed solution of the normal equations, but also to loss of information when the numerical rank of $\mathbf{A}$ is marginal like in the case of this example.

3.9 Introduction to Sparse Computing

3.9.1 About real life problems

Computational linear algebra is one of the most frequently used computational technique. Real life problems have a number of characteristics that can be taken into account when designing efficient solution algorithms involving linear algebra.

First of all, it is typical that realistic models lead to *large scale* problems. Storing and solving such problems require very powerful computing facilities. Sometimes even the most advanced computers cannot accommodate refined models of a given problem.

Second, when it comes to the linear algebra of the solution of large scale problems, the emerging vectors and matrices are usually sparse, often very sparse. *Sparsity* means that only a small proportion of all possible vector/matrix entries are different from zero. Let z denote the number of nonzeros in a vector or matrix. The *density* of an m dimensional vector $\mathbf{v}$ is defined as

$$\rho(\mathbf{v}) = \frac{z}{m}$$

Density of an $m \times n$ matrix $\mathbf{A}$ is:

$$\rho(\mathbf{A}) = \frac{z}{mn}$$

Sometimes density is expressed as percentage. Sparsity is nothing but low density.

There is an observation that in many cases even among the nonzero entries there are only few distinct values. For instance, it is typical that there are many ± 1 entries in a matrix. This phenomenon is called *super sparsity.* It is important to note that super sparsity is not the superlative of sparsity. It simply gives an additional characterization of the nature of sparsity. In some cases it is advantageous to utilize super sparsity to reduce memory requirements by storing each distinct value only once and using appropriate pointers to them.

Example 39 *Let* $\mathbf{v}$ *and* $\mathbf{A}$ *be*

$$\mathbf{v} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -3 \\ 1 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 4 & -1 \\ 0 & 0 & 0 & 3 & 1 \end{bmatrix}$$

In this case, $\rho(\mathbf{v}) = 3/5 = 0.6$ *(or 60%) and* $\rho(\mathbf{A}) = 10/25 = 0.4$ *(or 40%). Among the ten nonzeros of* $\mathbf{A}$ *there are five distinct values* $\{-1, 1, 2, 3, 4\}$.

Notions of 'small', 'medium' and 'large' as used to characterize the size of problems are just loosely defined. They keep changing over time as a result of developments in hardware, software, solution algorithms and computing techniques. In 1965 a 100 × 100 system of linear equations was large, now (at the beginning of the 21st century) it is considered small.

Typical densities of large problems ($m >$ 10,000) vary between 0.01% and 0.1%. Some researchers noticed that matrices often have no more than 10 nonzeros per column, independent of their size. It means that density decreases as problem size increases.

The main idea of sparse computing is to *store and perform calculations only on nonzeros.* The importance of the first part of this idea (storage) can be illustrated by the following example. Let us assume that in a 10,000×10,000 matrix each column has five nonzeros on the average, giving $\rho(\mathbf{A}) =$ 0.05%. If the matrix entries are stored as double precision (dp) numbers (the typical case) then the storage of the complete matrix would require 100,000,000 dp storage units which is 800Mb of memory. Not only the storage but the number of actual arithmetic operations would be prohibitively large. However, if only the nonzeros are stored, 50,000 dp units (0.4Mb) plus some administration are sufficient. This space is readily available on every desktop computer.

From experience, a third characteristic of large scale problems has been concluded: algorithms that work well for small problems are very often completely useless for medium and large problems.

The purpose of this section is to give an insight into sparse computing that makes all the difference in case of the solution of large scale real life problems.

The starting point is the storage of sparse entities (vectors and matrices). There are several different ways they can be stored. To make the correct choice, we have to consider the operations to be performed on them. There is an important distinction between *static* and *dynamic* data structures. The former remains unchanged during computations while the latter may change in size as a result of the computations (for instance, nonzeros can be created in place of original zeros during Gaussian elimination). Additionally, the size of a static data structure can be determined in advance. The size of a dynamic data structure is unknown and can grow substantially, sometimes even an estimate of the upper bound cannot be given.

3.9.2 Storing sparse vectors

The easiest way to store a sparse vector is to hold it in a *full-length* array. This is also called *explicit* storage of a vector. While this can be rather wasteful, the elements are directly accessible (by their subscript) and algorithms using such vectors are simple, though not too efficient if the vectors are really sparse.

As a better way, only the nonzeros of a sparse vector are stored as `(integer, double)` pairs (i, v_i), for $i \in Z$, where Z is the set of indices of nonzeros in $\mathbf{v}$. In a computer program we usually keep a separate integer and a double array each of length of the number of entries (at least). This is the *compressed* or *packed* storage of a sparse vector.

Example 40 *Vector* $\mathbf{v}$ *of Example 39 can be stored with* $Z = \{1, 4, 5\}$ *and* $|Z| = z = 3$ *as*

ind	1	4	5
val	1.0	-3.0	1.0

The storage is said to be *ordered* if the subscripts are in a monotone (usually ascending) order, like in this example, otherwise *unordered*. Usually, there is no need to maintain ordered form of sparse vectors (see later).

Sometimes we need to bring an explicit vector into packed form. This operation is called *gather*. The opposite operation is *scatter* when a packed vector is expanded into full-length representation.

3.9.3 Operations involving sparse vectors

Addition and accumulation of sparse vectors. In the Gaussian elimination a typical step (ERO–2) is the subtraction of a multiple of a row from another row (assuming $\mathbf{u}$ and $\mathbf{v}$ are row vectors now)

$$\mathbf{u} := \mathbf{u} - \lambda \mathbf{v}. \tag{66}$$

If both vectors are in unordered packed form then this operation would require search of vectors to identify the matching pairs and newly created nonzeros (fill-ins). A better way is the following. Assume there is a full-length work vector (array) $\mathbf{w}$ available with every position initialized to zero. We will work with this array, modify its elements, but at the end restore it to zero. One possible way is method-1:

1. Scatter $\mathbf{v}$ into $\mathbf{w}$.
2. Scan $\mathbf{u}$. For each entry u_i check w_i. If it is nonzero, update u_i ($u_i := u_i - \lambda w_i$) and reset w_i to zero.
3. Scan $\mathbf{v}$. For each entry v_i check w_i. If it is nonzero, we have a fill-in: $u_i := -\lambda w_i$ ($= -\lambda v_i$). Add the new component of $\mathbf{u}$ to the data structure of $\mathbf{u}$ and reset w_i to zero.

If the vectors are in ordered form then scan can be made without expansion but any fill-in (which is usually put to the end of the data structure) can destroy ordering in one step. This is one of the reasons why we do not assume ordering in packed form.

There is another way of performing (66). Here, again, we assume an explicit working vector $\mathbf{w} = \mathbf{0}$ is available. Method-2:

1. Scatter $\mathbf{u}$ into $\mathbf{w}$.
2. Scan $\mathbf{v}$. For each entry v_i check w_i. If it is nonzero, update w_i ($w_i := w_i - \lambda v_i$), otherwise set $w_i := -\lambda v_i$ and add i to the data structure of $\mathbf{u}$.
3. Scan the modified data structure of $\mathbf{u}$. For each i, set $u_i := w_i,\ w_i := 0$.

As with the previous case, at the end of this algorithm $\mathbf{w}$ is restored to zero and can be used for similar purposes again.

The second algorithm is slightly more expensive than the first one because the revised $\mathbf{u}$ is stored first in $\mathbf{w}$ and then placed in $\mathbf{u}$. However, if the following sequence of operations is to be performed

$$\mathbf{u} := \mathbf{u} + \sum_j \lambda_j \mathbf{v}_j,$$

then the second algorithm is more efficient. The reason is that steps 1 and 3 are performed only once and step 2 is repeated for $\mathbf{v}_1, \mathbf{v}_2, \ldots,$ with intermediate results for $\mathbf{u}$ remaining in $\mathbf{w}$.

Both algorithms share the following very important feature: *the number of operations to be performed is a function of the number of nonzeros and is completely independent of the explicit size (m) of the participating vectors.*

Dot product of two packed vectors. The dot product of two m dimensional vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as $\mathbf{u}^T\mathbf{v} = \sum_{i=1}^{m} u_i v_i$. With full-length vectors this operation is logically very simple. However, if the vectors are sparse, a large number of multiplications and additions may be performed with zero valued operands. If $\mathbf{u}$ and $\mathbf{v}$ are stored as sparse vectors, the best to do is to scatter one of them as described in section 3.9.2 and perform the dot product between a sparse and a full-length vector. If $\mathbf{u}$ is expanded into full representation in array `w` and `nzv` denotes the number of nonzeros in $\mathbf{v}$ then the following pseudo-code will do the job (assuming that the nonzeros of $\mathbf{v}$ are represented in the two arrays `indv` and `valv`).

```
dotprod = 0.0
do i = 1, nzv
  dotprod = dotprod + valv(i) * w(indv(i))
enddo
```

which requires one direct and one indirect access of arrays `valv` and `w`, respectively, in the body of the loop.

3.9.4 Storing sparse matrices

A sparse matrix can be considered as a set of sparse row or column vectors, or simply a set of nonzero elements together with their coordinates. We briefly discuss the different variants.

Collection of sparse column/row vectors. A straightforward way to store a sparse matrix is to store it as a collection of sparse column (row) vectors. The nonzeros of the columns (rows) can be stored consecutively in the same array, indices (row indices of entries in `rind`) and elements (in `val`) separately, of course. The starting position (beginning) of each column (say, `cbeg`) must be recorded. If we have n columns, we need $n+1$ such positions, the last one pointing to the first position after the last element of the last column. In this way the length (in terms of nonzeros) of column j is `clen(j) = cbeg(j+1) - cbeg(j)`. In general, `clen` is not needed explicitly. However, if the columns are not stored consecutively, it is necessary to maintain `clen`. In this case a column is accessed by `cbeg` and `clen`. Keeping `clen` makes the structure more flexible.

Table 1 shows how matrix $\mathbf{A}$ of Example 39 can be represented as a set of sparse column vectors. It is to be noted that some of the column vectors are unordered, that is, the indices of the elements are not in an ascending order. Under subscript 6, we enclosed 11 in parentheses to indicate that this entry of `cbeg` is needed only if `clen` is not used.

Subscripts	1	2	3	4	5	6	7	8	9	10
`cbeg`	1	3	4	5	8	(11)				
`clen`	2	1	1	3	3					
`rind`	3	1	3	2	4	1	5	3	4	5
`val`	1.0	−1.0	−1.0	1.0	4.0	2.0	3.0	2.0	−1.0	1.0

Table 1: Matrix $\mathbf{A}$ of Example 39 stored as a collection of sparse column vectors.

The main problem with this representation becomes evident when a new matrix element is to be inserted, that is, when this structure is used dynamically. If there is some free space left after each column the new elements can be placed there, if not, a fresh copy of the column is created and placed after the last up-to-date column leaving the out-of-date version unaltered. This action is possible only if the array has some 'elbow room' at the end. Since columns may not be consecutive anymore, both `cbeg` and `clen` are needed to access them. Unfortunately, in this way we can run out of the allocated array. Therefore, from time to time the vectors may have to be shifted up to form a consecutive sequence from the beginning again. This operation is called *compression*, more specifically, this is column compression.

Similar arguments can be used if $\mathbf{A}$ is stored as a collection of sparse row vectors.

Linked lists. The advantages of linked lists are well known in computer science. Sparse computing can also greatly benefit from using linked lists wherever it is appropriate.

We can store a matrix as a collection of *linked column (row) vectors*. Each column (row) has a *header* that points to the first element of the column (row). Additionally, each nonzero has a *pointer* pointing to the position of the next nonzero in the linked list. The last element points to null or any other invalid position. With linked lists, ordering within a column (row) is meaningless. The sequence of elements is determined by the way pointers have been set up.

Table 2 shows matrix $\mathbf{A}$ of Example 39 represented as a set of sparse linked column vectors. Here, `rlink` is the pointer to the next row index in the column. The elements of some of the columns are not held in consecutive memory locations.

If, during operations on the matrix, new elements have to be inserted into columns, we can place them to the end of the array and adjust the pointers accordingly (an exercise second year computing students are familiar with). It may happen that an element has to be deleted from the structure. This operation can be done efficiently if not only forward but also backward links are maintained within each column. In this case an additional header to the last element of each column is needed. The backward pointing linked list has the same properties as the forward list. Table 3 shows matrix $\mathbf{A}$ of Example 39 represented as a set of sparse doubly linked column vectors. Here, we use `cfhead` and `cbhead` to denote

Subscripts	1	2	3	4	5	6	7	8	9	10
`chead`	9	1	10	2	7					
`rind`	3	1	5	4	3	4	3	5	1	2
`val`	−1.0	2.0	3.0	4.0	1.0	−1.0	2.0	1.0	−1.0	1.0
`rlink`	0	4	0	3	0	8	6	0	5	0

Table 2: An instance of representing matrix **A** of Example 39 stored as a collection of sparse linked column vectors.

the column head of the forward and backward lists, respectively, while `rflink` and `rblink` stand for row forward and backward link, respectively.

Subscripts	1	2	3	4	5	6	7	8	9	10
`cfhead`	9	1	10	2	7					
`cbhead`	5	1	10	3	8					
`rind`	3	1	5	4	3	4	3	5	1	2
`val`	−1.0	2.0	3.0	4.0	1.0	−1.0	2.0	1.0	−1.0	1.0
`rflink`	0	4	0	3	0	8	6	0	5	0
`rblink`	0	0	4	2	9	7	0	6	0	0

Table 3: An instance of representing matrix **A** of Example 39 stored as a collection of sparse doubly linked column vectors.

As a summary, linked lists eliminate the need for ‘elbow room’ and column/row compression, they make insertion and deletion easy and efficient. This method has an increased memory requirement and will somewhat slow down matrix operations due to more indirect memory referencing. However, when the data structure is ‘very dynamic’ it is the method of choice.

Coordinate scheme. A coordinate scheme is obtained from a straightforward interpretation of matrices. Every element of a matrix is characterized by its position (row and column index) and actual value. In other words, if (i, j, a_{ij}) triplets are given for all nonzeros of an $m \times n$ matrix $\mathbf{A}$ then the matrix is uniquely defined. The elements need not be in any order.

While this scheme can be very convenient to prepare the matrix for input (it is a sort of free format), it is not particularly suitable for efficient operations involving matrices and vectors. If a matrix is given in this form, the usual practice is to convert it to one of the previously discussed representations (collection of sparse vectors or linked lists). It is possible to design a linear time algorithm (in terms of the number of nonzeros) to make the conversion. Details are omitted here.

Comparison of the schemes. From purely storage point of view, the three sparsity forms discussed above require the following amount of memory to represent an $m \times n$ matrix with z nonzeros. Here, we assume that a pointer is of the size of an integer.

Method	Integer	Double
Collection of sparse column vectors	$2n+z$	z
Collection of sparse row vectors	$2m+z$	z
Linked list by columns	$n+2z$	z
if doubly linked	$2n+3z$	z
Linked list by rows	$m+2z$	z
if doubly linked	$2m+3z$	z
Coordinate scheme	$2z$	z

The main question is: What is the critical density when the sparse storage becomes more practical than the explicit storage?

First, let us look at the speed of memory access. Experience shows that on a scalar processor, most operations involving one level indirect addressing are typically two to four times slower than the same operations using direct addressing. This suggests that the break even density is between 25% and 50%.

Next, let us compare the memory requirements of the different storage schemes. The explicit storage of an $m \times n$ matrix $\mathbf{A}$ requires mn dp locations ($8mn$ bytes). If the 'collection of sparse column vectors' scheme is used then $2n+z$ integers ($8n+4z$ bytes) and z dp's ($8z$ bytes), all together $8n+12z$ bytes are needed. Break even occurs when $8mn = 8n+12z$, from which, $z = \frac{2}{3}n(m-1)$. If we have this many nonzeros then the density is

$$\rho(\mathbf{A}) = \frac{\frac{2}{3}n(m-1)}{mn} = \frac{2}{3} \times \frac{m-1}{m}.$$

If m is large, the critical density for this case is about 67%.

For efficient sparse algorithms we have to consider both aspects, therefore, the critical density below which the sparse storage is more beneficial is $\min\{25\% - 50\%, 67\%\} = 25\% - 50\%$. We are definitely on the safe side if we take it 25%. It means that the first matrix storage scheme (collection of sparse column vectors) is more efficient for storage and operations if the density is 25% or less. Since we noted that matrices of real life problems are typically less than 1% dense, it is compelling to use sparse techniques in such cases.

Calculations for other sparse matrix storage schemes can be performed in a similar fashion to obtain the critical densities.

3.9.5 Operations involving sparse matrices

Matrix by vector products. Let us consider the $\mathbf{y} = \mathbf{A}\mathbf{x}$ product. If $\mathbf{x}$ is given as an explicit vector then any representation of $\mathbf{A}$ is suitable to perform the operation efficiently. If $\mathbf{x}$ is in sparse form then the preferred form for $\mathbf{A}$ is the collection of column vectors. Now we use the alternative definition of the product as $\mathbf{y} = x_1\mathbf{a}_1 + \cdots + x_n\mathbf{a}_n$ and perform operations with nonzero x_js only. This can be done most conveniently by method-2 of subsection 3.9.3.

Matrix by matrix products. Let $\mathbf{C} = \mathbf{AB}$ with $\mathbf{C} \in \mathbb{R}^{m\times n}$, $\mathbf{A} \in \mathbb{R}^{m\times p}$ and $\mathbf{B} \in \mathbb{R}^{p\times n}$. Recall that $c_{ij} = \mathbf{a}^i\mathbf{b}_j = \sum_k a_{ik}b_{kj}$. If we follow this formula then it is difficult to identify matching a_{ik}, b_{kj} pairs when both are $\neq 0$ since sparse vectors $\mathbf{a}^i$ and $\mathbf{b}_j$ may be unordered. Additionally, we cannot predict which c_{ij} positions will be nonzero, therefore, all possible mn positions have to be computed and only the nonzeros stored.

A better way is if the multiplication is performed by the alternative definition as $\mathbf{C} = \sum_{k=1}^{p} \mathbf{a}_k\mathbf{b}^k$, i.e., as a sum of p outer products made up of the columns of $\mathbf{A}$ and rows of $\mathbf{B}$. If a nonzero c_{ij} is generated in this way, it can be inserted into the data structure of $\mathbf{C}$ by some of the methods discussed. It is also clear that the preferred storage forms are: columnwise for $\mathbf{A}$ and rowwise for $\mathbf{B}$.

If both matrices are stored columnwise, $\mathbf{c}_j$ (column j of $\mathbf{C}$) can be accumulated as a linear combination of the columns of $\mathbf{A}$ by $\mathbf{c}_j = \sum_{k=1}^{p} b_{kj}\mathbf{a}_k$. This may be done conveniently by method-2 defined in subsection 3.9.3. Similar things can be done if both matrices are stored rowwise.

4 Convergent Sequences

This section presents some basic properties of convergent sequences. While there is an inherent relationship between series and sequences, the discussion will concentrate on the latter. First, we need the general definition of metric spaces that will be used in special cases.

4.1 Metric Spaces

Some important features of sequences depend on some properties of the distance between points and not on the fact that the points are in $\mathbb{R}^k$. When these properties are studied abstractly they lead to the concept of a metric space.

A ***metric space*** *is a nonempty set S of objects (called points) together with a function $d : S \times S \mapsto \mathbb{R}^1$ (called the metric of the space) satisfying the following four properties for all points $\mathbf{x}, \mathbf{y}, \mathbf{z} \in S$:*

1. $d(\mathbf{x}, \mathbf{x}) = 0$.
2. $d(\mathbf{x}, \mathbf{y}) > 0$ *if* $\mathbf{x} \neq \mathbf{y}$.
3. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ *(symmetry).*
4. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ *(triangle inequality).*

The nonnegative number $d(\mathbf{x}, \mathbf{y})$ is to be thought of as the distance from $\mathbf{x}$ to $\mathbf{y}$. In these terms the intuitive meaning of properties 1 through 4 is clear. A metric space is usually referred to as (S, d), where S is the set of points and d is the metric.

Example 41 *Let $S = \mathbb{R}^k$ and $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\,\|_2$. This is called the Euclidean metric.*

4.2 Limits

The *limit of a sequence* of real numbers $\{x_n\}$ is denoted symbolically by writing

$$\lim_{n \to \infty} x_n = A.$$

Its meaning, formally, is that for every $\epsilon > 0$ there is an integer N such that

$$|x_n - A| < \epsilon, \quad \text{whenever } n \geq N. \tag{67}$$

In practical terms it means that A is the limit of the sequence if x_n can be made arbitrarily close to A with sufficiently large n.

Example 42 *Let a sequence be defined by $x_n = \dfrac{n-1}{n}$. Its limit is 1, i.e., $\lim_{n\to\infty} x_n = 1$, because given any small positive ϵ, we can determine N such that (67) holds with $A = 1$. For instance, if $\epsilon = 0.001$ then we want to know N such that $\left|\dfrac{N-1}{N} - 1\right| < 0.001$. This can be solved for N giving $N = 1000$. It means that whenever $n > 1000$, the distance between x_n and 1 will be less than 0.001.*

There is also a *limit of a function* indicated by notation such as

$$\lim_{x \to p} f(x) = A, \tag{68}$$

which means that for every $\epsilon > 0$ there is another number $\delta > 0$ such that

$$|f(x) - A| < \epsilon \quad \text{whenever} \quad 0 < |x - p| < \delta.$$

Again, in practical terms it means that A is a limit of $f(x)$ at p if $f(x)$ can be made arbitrarily close to A by taking x sufficiently close to p.

Example 43 *Let $f(x) = x^3$ and $p = 3$. The limit of x^3 at $x = 3$ is 27. To see how definition (68) works in this case, let $\epsilon = 0.01$. Now $|x^3 - 27| < 0.01$ must hold. If this expression is solved for x we obtain that x must satisfy $2.9996 < x < 3.0004$ because $2.9996^3 \approx 26.99$ and $3.0004^3 \approx 27.01$. Hence, $\delta = 0.0004$. Here, an interesting observation can be made, namely, for a certain closeness in the function value, a much tighter closeness in the argument is needed at $p = 3$.*

On the other hand, function $f(x) = 1/x^2$ has no limit at $x = 0$ since it can be made arbitrarily large in the neighbourhood of 0.

What was just said about limits of sequences of numbers and functions of a single variable can be easily generalized for points (vectors) in higher dimensional Euclidean spaces $\mathbb{R}^k$, $k > 1$. To be able to do so, a measure for the distance between two points is needed. Any of the discussed norms can be used for this purpose by saying that the distance between $\mathbf{x}$ and $\mathbf{y}$ is $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$.

A sequence $\{\mathbf{x}_n\}$ of points in a (vector) space S is said to *converge* if there is a point $\mathbf{p}$ in S with the following property: For every $\epsilon > 0$ there is an integer N such that

$$d(\mathbf{x}_n, \mathbf{p}) < \epsilon \quad \text{whenever} \quad n \geq N.$$

We also say that $\{\mathbf{x}_n\}$ converges to $\mathbf{p}$ and we write $\mathbf{x}_n \to \mathbf{p}$ as $n \to \infty$, or simply $\mathbf{x}_n \to \mathbf{p}$. If there is no such $\mathbf{p}$ in S, the sequence $\{\mathbf{x}_n\}$ is said to diverge. This definition of convergence implies that

$$\mathbf{x}_n \to \mathbf{p} \quad \text{if and only if} \quad d(\mathbf{x}_n, \mathbf{p}) \to 0.$$

The convergence of the sequence of distances $\{d(\mathbf{x}_n, \mathbf{p})\}$ to 0 takes place in the one dimensional Euclidean space $\mathbb{R}^1$.

In Euclidean space $\mathbb{R}^1$, a sequence $\{x_n\}$ is called *increasing* if $x_n \leq x_{n+1}$ for all n. If an increasing sequence is bounded above (that is, if $x_n \leq M$ for some $M < +\infty$ and all n), then $\{x_n\}$ is convergent. For example, $(n-1)/n$ converges to 1. Similarly, $\{x_n\}$ is called *decreasing* if $x_n \geq x_{n+1}$ for all n. Every decreasing sequence which is bounded below converges. For example, $1/n$ converges to 0.

If $\{a_n\}$ and $\{b_n\}$ are sequences of nonnegative real numbers converging to 0, then $\{a_n + b_n\}$ also converges to 0. If $0 \leq c_n \leq a_n$ for all n and if $\{a_n\}$ converges to 0, then $\{c_n\}$ also converges to 0. These elementary properties of sequences in $\mathbb{R}^1$ can be used to simplify some of the proofs concerning limits in higher dimensional Euclidean spaces. As an illustration, let us consider the following important theorem.

Theorem 2 *A sequence $\{\mathbf{x}_n\}$ in $\mathbb{R}^k$ can converge to at most one point in $\mathbb{R}^k$.*

Proof. Assume that $\mathbf{x}_n \to \mathbf{p}$ and $\mathbf{x}_n \to \mathbf{q}$. We will prove that $\mathbf{p} = \mathbf{q}$. By the triangle inequality of the norm (see page 11) which is the distance now, we have

$$0 \leq d(\mathbf{p}, \mathbf{q}) \leq d(\mathbf{p}, \mathbf{x}_n) + d(\mathbf{x}_n, \mathbf{q}).$$

Since $d(\mathbf{p}, \mathbf{x}_n) \to 0$ and $d(\mathbf{x}_n, \mathbf{q}) \to 0$ this implies that $d(\mathbf{p}, \mathbf{q}) = 0$, so $\mathbf{p} = \mathbf{q}$. ■

If a sequence $\{\mathbf{x}_n\}$ converges, the unique point to which it converges is called the *limit* of the sequence and is denoted by $\lim \mathbf{x}_n$ or by $\lim_{n\to\infty} \mathbf{x}_n$.

In the one dimensional Euclidean space we have $\lim_{n\to\infty} 1/n = 0$. The same sequence in the subspace $T = (0, 1]$ does not converge because the only candidate for the limit is 0 and $0 \notin T$. This example shows that the convergence or divergence of a sequence depends on the underlying space as well as on the definition of the distance.

4.3 Cauchy Sequences

If a sequence $\{\mathbf{x}_n\}$ converges to a limit $\mathbf{p}$, its terms must ultimately become close to $\mathbf{p}$ and hence close to each other. This property is stated more formally in the next theorem.

Theorem 3 *Assume that $\{\mathbf{x}_n\}$ converges in $\mathbb{R}^k$. Then for every $\epsilon > 0$ there is an integer N such that*

$$d(\mathbf{x}_n, \mathbf{x}_m) < \epsilon \quad \textit{whenever} \;\; n \geq N \textit{ and } \; m \geq N.$$

Proof. Let $\mathbf{p} = \lim \mathbf{x}_n$. Given $\epsilon > 0$, let N be such that $d(\mathbf{x}_n, \mathbf{p}) < \epsilon/2$ whenever $n \geq N$. Then $d(\mathbf{x}_m, \mathbf{p}) < \epsilon/2$ if $m \geq N$. If both $n \geq N$ and $m \geq N$ the triangle inequality gives us

$$d(\mathbf{x}_n, \mathbf{x}_m) \leq d(\mathbf{x}_n, \mathbf{p}) + d(\mathbf{p}, \mathbf{x}_m) < \epsilon/2 + \epsilon/2 = \epsilon$$

which completes the proof. ■

This theorem intuitively means that for any arbitrary precision (ϵ), we can find an index (N) beyond which any two elements of the convergent sequence are closer to each other than the given precision.

Definition of a Cauchy Sequence. *A sequence $\{\mathbf{x}_n\}$ in $\mathbb{R}^k$ is called a Cauchy sequence if it satisfies the following condition (called the Cauchy condition):*

For every $\epsilon > 0$ there is an integer N such that

$$d(\mathbf{x}_n, \mathbf{x}_m) < \epsilon \quad \textit{whenever} \;\; n \geq N \textit{ and } m \geq N.$$

Theorem 3 states that every convergent sequence is a Cauchy sequence. The converse is also true in $\mathbb{R}^k$ as it is stated in the following theorem which is given without proof.

Theorem 4 *In Euclidean space $\mathbb{R}^k$ every Cauchy sequence is convergent.*

This theorem is often used for proving the convergence of a sequence when the limit is not known in advance.

Example 44 *Consider the sequence $\{x_n\}$ in $\mathbb{R}^1$ defined by*

$$x_n = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{(-1)^{n-1}}{n}.$$

If $m > n \geq N$, we find (by taking successive terms in pairs) that

$$|x_m - x_n| = \left| \frac{1}{n+1} - \frac{1}{n+2} + \cdots \pm \frac{1}{m} \right| < \frac{1}{n} \leq \frac{1}{N},$$

so $|x_m - x_n| < \epsilon$ as soon as $N > 1/\epsilon$. Therefore, $\{x_n\}$ is a Cauchy sequence and hence it converges to some limit. It can be shown that this limit is $\log 2$, *a fact which is not immediately obvious.*

Example 45 *As another example, let us consider the following. Given a sequence $\{a_n\}$ of real numbers such that $|a_{n+2} - a_{n+1}| \leq \frac{1}{2}|a_{n+1} - a_n|$ for all $n \geq 1$. We can prove that $\{a_n\}$ converges without knowing its limit by showing it is a Cauchy sequence. The technique used here is important for the proof of the fixed-point theorem.*

Let $b_n = |a_{n+1} - a_n|$. Then $0 \leq b_{n+1} \leq b_n/2$. So, by induction, $b_{n+1} \leq b_1/2^n$. Hence $b_n \to 0$. Also, if $m > n$ we have

$$a_m - a_n = \sum_{k=n}^{m-1} (a_{k+1} - a_k),$$

hence

$$\begin{aligned}
|a_m - a_n| &= \left|\sum_{k=n}^{m-1}(a_{k+1}-a_k)\right| \\
&\leq \sum_{k=n}^{m-1}|a_{k+1}-a_k| \\
&= b_n + b_{n+1} + b_{n+2} + \cdots + b_{m-1} \\
&\leq b_n + \frac{b_n}{2} + \frac{b_n}{2^2} + \cdots + \frac{b_n}{2^{m-1-n}} \\
&= b_n\left(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^{m-1-n}}\right) \\
&< 2b_n \to 0.
\end{aligned}$$

This implies that $\{a_n\}$ is a Cauchy sequence, therefore it converges.

4.4 Fixed-point Theorem for Contractions

In this subsection we present the background of some general computational techniques for solving linear and nonlinear equations. Some examples will illustrate the findings.

First, we define the *complete metric spaces.* A metric space (S, d) is called complete if every Cauchy sequence in S converges in S, i.e., the limit point is also in S. A subset T of S is called complete if the metric subspace (T, d) is complete. For example, every Euclidean space $\mathbb{R}^k$ is complete. In particular, $\mathbb{R}^1$ is complete, but the subspace $T = (0, 1]$ is not complete.

Now, let $\mathbf{f} : S \mapsto S$ be a function from a metric space into itself. A point $\mathbf{p}$ in S is called a *fixed point* of $\mathbf{f}$ if $\mathbf{f}(\mathbf{p}) = \mathbf{p}$. The function $\mathbf{f}$ is called a *contraction* of S if there is a number α, $0 < \alpha < 1$ (called a *contraction constant*), such that

$$d(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y})) \leq \alpha d(\mathbf{x}, \mathbf{y}) \quad \text{for all} \quad \mathbf{x}, \mathbf{y} \in S. \tag{69}$$

Theorem 5 (Fixed-Point theorem) *A contraction* $\mathbf{f}$ *of a complete metric space S has a unique fixed point* $\mathbf{p}$.

It is quite instructive to see the logic of the proof.

Proof. If $\mathbf{p}$ and $\mathbf{p}'$ are two fixed points, (69) implies $d(\mathbf{p}, \mathbf{p}') = d(f(\mathbf{p}), f(\mathbf{p}')) \leq \alpha d(\mathbf{p}, \mathbf{p}')$, so $d(\mathbf{p}, \mathbf{p}') = 0$ and $\mathbf{p} = \mathbf{p}'$. Hence $\mathbf{f}$ has at most one fixed point.

To prove that it has one, take any point $\mathbf{x}$ in S and consider the sequence of iterates:

$$\mathbf{x},\ \mathbf{f}(\mathbf{x}),\ \mathbf{f}(\mathbf{f}(\mathbf{x})), \ldots$$

That is, define a sequence $\{\mathbf{p}_n\}$ intuitively as follows:

$$\mathbf{p}_0 = \mathbf{x}, \quad \mathbf{p}_{n+1} = \mathbf{f}(\mathbf{p}_n), \quad n = 0, 1, 2, \ldots$$

We will prove that $\{\mathbf{p}_n\}$ converges to a fixed point of $\mathbf{f}$. First, we show that $\{\mathbf{p}_n\}$ is a Cauchy sequence. From (69) we have

$$d(\mathbf{p}_{n+1}, \mathbf{p}_n) = d(\mathbf{f}(\mathbf{p}_n), \mathbf{f}\,(\mathbf{p}_{n-1})) \leq \alpha d(\mathbf{p}_n, \mathbf{p}_{n-1}),$$

so, by induction, we find

$$d(\mathbf{p}_{n+1}, \mathbf{p}_n) \leq \alpha^n d(\mathbf{p}_1, \mathbf{p}_0) = c\alpha^n,$$

where $c = d(\mathbf{p}_1, \mathbf{p}_0)$. Using the triangle inequality, we find for $m > n$ that

$$d(\mathbf{p}_m, \mathbf{p}_n) \leq \sum_{k=n}^{m-1} d(\mathbf{p}_{k+1}, \mathbf{p}_k) \leq c \sum_{k=n}^{m-1} \alpha^k = c\frac{\alpha^n - \alpha^m}{1-\alpha} < \frac{c}{1-\alpha}\alpha^n.$$

Since $\alpha^n \to 0$ as $n \to \infty$, this inequality shows that $\{\mathbf{p}_n\}$ is a Cauchy sequence. But S is complete, so there is a point $\mathbf{p}$ in S such that $\mathbf{p}_n \to \mathbf{p}$. By continuity of $\mathbf{f}$,

$$\mathbf{f}(\mathbf{p}) = \mathbf{f}\left(\lim_{n\to\infty} \mathbf{p}_n\right) = \lim_{n\to\infty} \mathbf{f}(\mathbf{p}_n) = \lim_{n\to\infty} \mathbf{p}_{n+1} = \mathbf{p}.$$

so $\mathbf{p}$ is a fixed point of $\mathbf{f}$. This completes the proof. ■

4.5 Iterative Methods for $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A}$ Square

The solution methods for $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{m\times m}$, considered so far are called *direct methods.* They compute the exact answer after m steps (in the absence of roundoff error). As to the arithmetic operations, the Gaussian elimination and the Cholesky factorization require $O(\frac{1}{3}m^3)$ and $O(\frac{1}{6}m^3)$ flops, respectively.

In many real-life problems m, the size of $\mathbf{A}$, is in the thousands or even in the tens of thousands. To solve problems of this size with direct methods is practically impossible. However, the emerging matrices tend to be very sparse and the number of nonzeros is usually in a manageable order of magnitude.

If elimination steps are applied to a sparse matrix, it is typical that, as a result of the transformations, positions which were zeros will become nonzeros. This phenomenon is called *fill-in.* Fill-in can be so heavy that the whole triangular matrix may become full. So, the advantages of sparsity can disappear during the elimination process. For such cases something better is needed.

As opposed to direct methods, there are *iterative methods* for solving $\mathbf{Ax} = \mathbf{b}$. They start with an approximate solution $\mathbf{x}^{(0)}$ and generate a sequence of approximations $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}, \ldots$ such that $\{\mathbf{x}^{(k)}\}$ converges to the solution $\mathbf{x}$ of the original system. The iterative methods have the advantage that they work with fixed matrices throughout and, therefore, do not generate fill-in. As such, they are ideally suited for solving large sparse problems that otherwise would produce heavy fill-in with a direct method.

4.5.1 Paradigm of iterative methods

A *splitting* of a square matrix $\mathbf{A}$ is defined as $\mathbf{M} = \mathbf{A} - \mathbf{N}$ where $\mathbf{M}$ is a nonsingular matrix. Splitting yields an iterative method as follows. Rewriting $\mathbf{Ax} = \mathbf{b}$, we have: $\mathbf{Ax} = \mathbf{Mx} + \mathbf{Nx} = \mathbf{b}$, which implies $\mathbf{Mx} = \mathbf{b} - \mathbf{Nx}$ and $\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} - \mathbf{M}^{-1}\mathbf{Nx}$. Denoting $\mathbf{G} = -\mathbf{M}^{-1}\mathbf{N}$ and $\mathbf{c} = \mathbf{M}^{-1}\mathbf{b}$, we get $\mathbf{x} = \mathbf{Gx} + \mathbf{c}$ and can take

$$\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c} \tag{70}$$

as our iterative method. Based on our knowledge of convergent sequences, we can investigate the convergence of the sequence generated by (70).

Theorem 6 *Let $\|\cdot\|$ be any matrix norm. If $\|\mathbf{G}\| < 1$, then $\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}$ converges for any starting point $\mathbf{x}^{(0)}$.*

Proof. Subtract $\mathbf{x} = \mathbf{Gx} + \mathbf{c}$ from $\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}$ to get $\mathbf{x}^{(k+1)} - \mathbf{x} = \mathbf{G}(\mathbf{x}^{(k)} - \mathbf{x})$. For the norm: $\|\mathbf{x}^{(k+1)} - \mathbf{x}\| \leq \|\mathbf{G}\|\,\|\mathbf{x}^{(k)} - \mathbf{x}\| \leq \|\mathbf{G}\|^2\,\|\mathbf{x}^{(k-1)} - \mathbf{x}\| \leq \cdots \leq \|\mathbf{G}\|^{k+1}\|\mathbf{x}^{(0)} - \mathbf{x}\|$, which converges to 0 since $\|\mathbf{G}\| < 1$. ■

Let λ_i, $i = 1, \ldots, m$, denote the eigenvalues of $\mathbf{G}$. The *spectral radius* of $\mathbf{G}$ is $\rho(\mathbf{G}) = \max_i\{|\lambda_i|\}$.

Theorem 7 *Let $\mathbf{G}$ be defined as above. Any of the following conditions is sufficient for the convergence of (70):*

1. *$\lim_{k\to\infty} \mathbf{G}^k = \mathbf{0}$.*
2. *$\lim_{k\to\infty} \mathbf{G}^k\mathbf{x} = \mathbf{0}, \quad \forall \mathbf{x} \in \mathbb{R}^m$.*
3. *$\rho(\mathbf{G}) < 1$.*

Proof is omitted here. The important thing is that we have three more sufficient criteria to ensure convergence. If any of them can be verified for a given problem then the iterative method will converge.

The next issue of interest is the rate (speed) of the convergence. The *rate of convergence* of $\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}$ is defined as $r(\mathbf{G}) = -\log_{10}\rho(\mathbf{G})$. This can be interpreted as the increase in the number of correct decimal places in the solution per iteration. The smaller is $\rho(\mathbf{G})$ the higher is the rate of convergence, i.e., the greater is the number of correct decimal places computed per iteration.

(70) defines a family of iterative methods. The actual choice of $\mathbf{G}$ and $\mathbf{c}$ results in specific computational algorithms. Our goal is to choose a splitting $\mathbf{A} = \mathbf{M} + \mathbf{N}$ so that both

1. $\mathbf{G}\mathbf{x} = -\mathbf{M}^{-1}\mathbf{N}\mathbf{x}$ and $\mathbf{c} = \mathbf{M}^{-1}\mathbf{b}$ are easy to compute, and
2. $\rho(\mathbf{G})$ is small.

These are conflicting requirements and we need to find some compromise between the following extreme cases:

- $\mathbf{M} = \mathbf{I}$ serves goal 1 extremely well but may not make $\rho(\mathbf{G}) < 1$.
- $\mathbf{M} = \mathbf{A}$ and $\mathbf{N} = \mathbf{0}$ are good for goal 2 but probably bad for 1.

Some of the most important iterative methods for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ are discussed in the subsequent sections. The methods share the following notations. Assuming that the diagonal of $\mathbf{A}$ is zero free, we write

$$\mathbf{A} = \mathbf{D} - \tilde{\mathbf{L}} - \tilde{\mathbf{U}} = \mathbf{D}(\mathbf{I} - \mathbf{L} - \mathbf{U}),$$

where $\mathbf{D}$ is the diagonal of $\mathbf{A}$, $-\tilde{\mathbf{L}}$ is the strictly lower triangular part of $\mathbf{A}$ with $\mathbf{D}\mathbf{L} = \tilde{\mathbf{L}}$ and $-\tilde{\mathbf{U}}$ is the strictly upper triangular part of $\mathbf{A}$ with $\mathbf{D}\mathbf{U} = \tilde{\mathbf{U}}$.

Example 46

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

from where

$$\mathbf{D} = \begin{bmatrix} 2 & & \\ & 1 & \\ & & 2 \end{bmatrix}, \quad \tilde{\mathbf{L}} = \begin{bmatrix} 0 & & \\ 3 & 0 & \\ -6 & 2 & 0 \end{bmatrix}, \quad \tilde{\mathbf{U}} = \begin{bmatrix} 0 & 4 & -2 \\ & 0 & 5 \\ & & 0 \end{bmatrix}$$

and because $\mathbf{L} = \mathbf{D}^{-1}\tilde{\mathbf{L}}$ and $\mathbf{U} = \mathbf{D}^{-1}\tilde{\mathbf{U}}$, we have

$$\mathbf{L} = \begin{bmatrix} 0 & & \\ 3 & 0 & \\ -3 & 1 & 0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 0 & 2 & -1 \\ & 0 & 5 \\ & & 0 \end{bmatrix}.$$

Finally:

$$\mathbf{A} = \begin{bmatrix} 2 & -4 & 2 \\ -3 & 1 & -5 \\ 6 & -2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & & \\ & 1 & \\ & & 2 \end{bmatrix} \left(\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} - \begin{bmatrix} 0 & & \\ 3 & 0 & \\ -3 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 2 & -1 \\ & 0 & 5 \\ & & 0 \end{bmatrix} \right)$$

4.5.2 Jacobi method

Assuming that the diagonal of $\mathbf{A}$ is zero free means $\mathbf{D}$ is nonsingular. The Jacobi method is defined as follows. Let $\mathbf{M} = \mathbf{D} = \operatorname{diag}(a_{11}, a_{22}, \ldots, a_{mm})$ and $\mathbf{N} = \mathbf{A} - \mathbf{D}$. In this case $\mathbf{D}^{-1} = \operatorname{diag}(1/a_{11}, 1/a_{22}, \ldots, 1/a_{mm})$. Now, $\mathbf{Ax} = \mathbf{b}$ becomes $\mathbf{Dx} + \mathbf{Nx} = \mathbf{b}$. Matrices $\mathbf{D}$ and $\mathbf{N}$ are

$$\mathbf{D} = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{mm} \end{bmatrix} \quad \text{and} \quad \mathbf{N} = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1m} \\ a_{21} & 0 & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & 0 \end{bmatrix}.$$

Multiplying by $\mathbf{D}^{-1}$ and rearranging, we obtain $\mathbf{x} = \mathbf{D}^{-1}\mathbf{b} - \mathbf{D}^{-1}\mathbf{Nx}$. Denoting $\mathbf{G} = -\mathbf{D}^{-1}\mathbf{N}$ and $\mathbf{c} = \mathbf{D}^{-1}\mathbf{b}$, we obtain $\mathbf{x} = \mathbf{Gx} + \mathbf{c}$ from which the following sequence can be defined:

$$\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}, \quad k = 0, 1, 2, \ldots \quad \text{with some choice of } \mathbf{x}^{(0)}.$$

From this, we obtain the following expression for row i:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right).$$

Calculating the new value of x_i requires access only to row i of the matrix. This simple fact suggests that this method is extremely suitable for parallel computing since data and work can be distributed well and little communication is needed per iteration.

4.5.3 Gauss-Seidel method

The motivation for this method is that at the ith step of an iteration of the Jacobi method, we have the improved values of the first $i-1$ components of the solution, so we can use them in the current sum. This modification is the Gauss-Seidel method which has the following general step in iteration k:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \underbrace{\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)}}_{\text{updated } x\text{'s}} - \underbrace{\sum_{j=i+1}^{m} a_{ij} x_j^{(k)}}_{x\text{'s of previous iteration}} \right) \tag{71}$$

To derive the matrix form of this method, first (71) is rearranged:

$$\sum_{j=1}^{i} a_{ij} x_j^{(k+1)} = - \sum_{j=i+1}^{m} a_{ij} x_j^{(k)} + b_i.$$

This can further be rewritten as $(\mathbf{D} - \tilde{\mathbf{L}})\mathbf{x}^{(k+1)} = \tilde{\mathbf{U}}\mathbf{x}^{(k)} + \mathbf{b}$, or

$$\begin{aligned} \mathbf{x}^{(k+1)} &= (\mathbf{D} - \tilde{\mathbf{L}})^{-1}\tilde{\mathbf{U}}\mathbf{x}^{(k)} + (\mathbf{D} - \tilde{\mathbf{L}})^{-1}\mathbf{b} \\ &= (\mathbf{I} - \mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{I} - \mathbf{L})^{-1}\mathbf{D}^{-1}\mathbf{b} \\ &= \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}. \end{aligned}$$

The Gauss-Seidel method is not specifically suitable for parallel implementation (look at (71)).

4.5.4 Successive over-relaxation, SOR(ω)

While the Gauss-Seidel (G-S) method was an improvement of the Jacobi method, SOR(ω) is a generalization of the G-S. This is achieved by taking an appropriate weighted average of the newly computed $x_i^{(k+1)}$ and the previous $x_i^{(k)}$:

$$x_i^{(k+1)} := (1-\omega)x_i^{(k)} + \omega x_i^{(k+1)},$$

where ω is called the *relaxation parameter* and ':=' is the assignment operator.

Step i of an SOR(ω) iteration is

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{m} a_{ij}x_j^{(k)}\right).$$

Again, to obtain the matrix form of the iterative scheme, we first rearrange the above equation:

$$a_{ii}x_i^{(k+1)} + \omega\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} = (1-\omega)a_{ii}x_i^{(k)} - \omega\sum_{j=i+1}^{m} a_{ij}x_j^{(k)} + \omega b_i$$

and write

$$(\mathbf{D} - \omega\tilde{\mathbf{L}})\mathbf{x}^{(k+1)} = ((1-\omega)\mathbf{D} + \omega\tilde{\mathbf{U}})\mathbf{x}^{(k)} + \omega\mathbf{b},$$

which gives

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= (\mathbf{D} - \omega\tilde{\mathbf{L}})^{-1}((1-\omega)\mathbf{D} + \omega\tilde{\mathbf{U}})\mathbf{x}^{(k)} + \omega(\mathbf{D} - \omega\tilde{\mathbf{L}})^{-1}\mathbf{b} \\
&= (\mathbf{I} - \omega\mathbf{L})^{-1}((1-\omega)\mathbf{I} + \omega\mathbf{U})\mathbf{x}^{(k)} + \omega(\mathbf{I} - \omega\mathbf{L})^{-1}\mathbf{D}^{-1}\mathbf{b} \\
&= \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}.
\end{aligned}$$

Depending on the value of ω, we distinguish three cases: $\omega = 1$ is equivalent to the G-S method, $\omega < 1$ is called *underrelaxation*, and $\omega > 1$ is called *overrelaxation.* Useful values for ω are discussed in the next section.

4.5.5 Convergence of the methods

In this section, a number of criteria are given that guarantee the convergence of the iterative methods just discussed. To present them, some definitions are needed.

A square matrix $\mathbf{A}$ is *strictly row diagonally dominant* if $|a_{ii}| > \sum_{j\neq i}|a_{ij}|$ for all i. A square matrix $\mathbf{A}$ is *weakly row diagonally dominant* if $|a_{ii}| \geq \sum_{j\neq i}|a_{ij}|$ for all i, with strict inequality at least once.

There will be a criterion which refers to a special nonzero pattern of the matrix. A square matrix $\mathbf{A}$ is said to be *irreducible* if, by symmetric permutation of rows and columns, it cannot be brought to the following partitioned form

$$\left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{0} & \mathbf{A}_{22} \end{array}\right],$$

where $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are square submatrices.

Now, we state the results (proofs omitted except one case) relevant to our iterative methods.

1. If $\mathbf{A}$ is strictly row diagonally dominant then the Jacobi and the G-S methods both converge, and the G-S method is faster. The advantage of this criterion is that it is easy to check. However, it is not always applicable. It is important to note that this is a sufficient condition only. There can be cases when this is not satisfied but the method still converges. Therefore, if it fails, it is worth checking some other conditions.

   In case of the Jacobi method the statement is: If $\mathbf{A}$ of $\mathbf{Ax} = \mathbf{b}$ is a strictly row diagonally dominant matrix then the Jacobi method converges. It can be proved in the following way.

We prove that $\|\mathbf{G}\|_\infty < 1$ which implies convergence.

Reminder: in case of Jacobi $\mathbf{G} = -\mathbf{D}^{-1}\mathbf{N}$,

$$\begin{aligned}
\mathbf{G} &= -\begin{bmatrix} 1/a_{11} & & & \\ & 1/a_{22} & & \\ & & \ddots & \\ & & & 1/a_{mm} \end{bmatrix}\begin{bmatrix} 0 & a_{12} & \dots & a_{1m} \\ a_{21} & 0 & \dots & a_{2m} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & 0 \end{bmatrix} \\
&= -\begin{bmatrix} 0 & \dfrac{a_{12}}{a_{11}} & \dots & \dfrac{a_{1m}}{a_{11}} \\ \dfrac{a_{21}}{a_{22}} & 0 & \dots & \dfrac{a_{2m}}{a_{22}} \\ \vdots & & \ddots & \vdots \\ \dfrac{a_{m1}}{a_{mm}} & \dfrac{a_{m2}}{a_{mm}} & \dots & 0 \end{bmatrix}
\end{aligned}$$

But, strict row diagonal dominance means that for every row

$$|a_{i1}| + \dots + |a_{i,i-1}| + |a_{i,i+1}| + \dots + |a_{im}| < |a_{ii}|.$$

Dividing both sides by $|a_{ii}| > 0$ we obtain

$$\sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} = \sum_{j \neq i} \left| \frac{a_{ij}}{a_{ii}} \right| < 1.$$

It means the absolute row sum of $\mathbf{G}$ is < 1 for each row, thus it also holds for the largest sum which is the ℓ_∞ norm of $\mathbf{G}$. Therefore, $\|\mathbf{G}\|_\infty < 1$, thus the sequence generated by the Jacobi method converges.

2. If $\mathbf{A}$ is weakly row diagonally dominant and is irreducible then the Jacobi and the G-S methods both converge, and again, the G-S method is faster.

3. In case of SOR(ω), $0 < \omega < 2$ is necessary for the convergence. If $\mathbf{A}$ is positive definite this condition is also sufficient for the convergence.

There is one more open question to be answered: What is a/the stopping criterion of these iterative methods? In other words: When can we say that the solution is good enough?

Usually, we are satisfied with a good approximation instead of the (sometimes hopeless) exact solution. Therefore, if subsequent iterations make little change in $\mathbf{x}^{(k)}$ then we can stop. To be more specific, we prescribe a certain level of accuracy, say δ, and stop the iterations when for the absolute error $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \delta$. This criterion is problem and scale dependent. A much better criterion is to stop if

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|}{1 + \|\mathbf{x}^{(k)}\|} < \delta,$$

an expression similar to the mixed error measure. Typical values for δ are $10^{-10} \le \delta \le 10^{-6}$, depending on the desired relative accuracy.

4.5.6 Iterative refinement of a 'direct' solution

Detailed error analysis of the Gaussian elimination (not presented here) shows that the error of the computed solution of $\mathbf{Ax} = \mathbf{b}$ can be as large as $\varepsilon\kappa(\mathbf{A})$, where $\kappa(\mathbf{A}) = \mathrm{cond}(\mathbf{A})$ is the condition number of $\mathbf{A}$ and ε is the relative accuracy of the number representation which is $\approx 10^{-16}$ in case of IEEE double precision arithmetic. It implies that the computed solution (which is now denoted by $\mathbf{x}^{(0)}$) may not be accurate enough. If $\kappa(\mathbf{A})$ is not extremely large then $\mathbf{x}^{(0)}$ can be taken as the starting approximate solution which we can try to improve. In such a case the following iterative procedure, for $k = 0, 1, 2, \ldots$, will help.

1. Compute $\mathbf{r} = \mathbf{Ax}^{(k)} - \mathbf{b}$.
2. Solve $\mathbf{Ad} = \mathbf{r}$ for $\mathbf{d}$.
3. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{d}$ as the new approximation.

In step 2, a system of equations has to be solved involving matrix $\mathbf{A}$. If we used the LU factorization form of the Gaussian elimination to solve $\mathbf{Ax} = \mathbf{b}$ in the first place then solving $\mathbf{Ad} = \mathbf{r}$ in step 2 requires a forward and a back substitution only.

Under some relatively mild conditions (not presented here) this method can improve the solution quite considerably in very few $(2 - 10)$ and cheap iterations. The stopping criterion can be determined in a similar way as for other iterative methods discussed above.

5 Functions of Several Variables

There are functional relationships where several independent variables determine the value of the function. For instance, we can consider the temperature in an arena as a function of the three space coordinates, like $t = f(x, y, z)$.

More generally, we can have n independent variables, $x_1, \ldots, x_n$, that determine the function. This is written as $f(\mathbf{x}) = f(x_1, \ldots, x_n)$. Function f may be defined only on a subset of $\mathbb{R}^n$. We assume that the value of a function is a real number and say that f is a *real-valued function*.

A real-valued function f defined on a subset of $\mathbb{R}^n$ is said to be *continuous* at $\mathbf{x}$ if $\mathbf{x}^{(k)} \to \mathbf{x}$ implies $f(\mathbf{x}^{(k)}) \to f(\mathbf{x})$. Equivalently, f is continuous at $\mathbf{x}$ if for any given $\epsilon > 0$ there is a $\delta > 0$ such that $\|\mathbf{y} - \mathbf{x}\| < \delta$ implies $|f(\mathbf{y}) - f(\mathbf{x})| < \epsilon$, i.e., for points $\mathbf{y}$ close enough to $\mathbf{x}$ the function values $f(\mathbf{y})$ and $f(\mathbf{x})$ are also close (within a predetermined arbitrary accuracy ϵ).

5.1 Partial Differentiation, The Gradient, The Hessian

Let $f(\mathbf{x}) = f(x_1, \ldots, x_n)$. If $n-1$ variables are kept constant and just one is allowed to vary then we have a function of one variable. If this function is differentiable we can determine its derivative in the usual way. Let x_i be the selected variable. The partial differentiation of f with respect to x_i is denoted by $\dfrac{\partial f(\mathbf{x})}{\partial x_i}$. The C^1 function class is defined to be the set of functions that have continuous partial derivatives with respect to all variables.

Example 47 *Let $f(x, y, z) = x^2 - 2xyz + y^2 + z^2/2$. The partial derivatives are:*

$$\begin{aligned}
\frac{\partial f(x,y,z)}{\partial x} &= 2x - 2yz \quad \textit{(everything kept constant except } x\textit{)},\\
\frac{\partial f(x,y,z)}{\partial y} &= -2xz + 2y,\\
\frac{\partial f(x,y,z)}{\partial z} &= -2xy + z.
\end{aligned}$$

If $f \in C^1$ is a real-valued function on $\mathbb{R}^n$, $f(\mathbf{x}) = f(x_1, \ldots, x_n)$, the *gradient* of f is defined to be the row vector

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right].$$

It is important to remember that in matrix calculations the gradient is always considered to be a row vector and the transpose sign is omitted.

Though the entries of the gradient are functions, for any given $\mathbf{x}$ they determine a row vector of numerical values.

Example 48 *Let* $f(\mathbf{x}) = (x_1 - 2x_2)^3 - 3x_3^3$. *Then*

$$\nabla f(\mathbf{x}) = \left[3(x_1 - 2x_2)^2,\ -6(x_1 - 2x_2)^2,\ -9x_3^2\right].$$

At point $\mathbf{x} = [1, 1, -1/3]^T$ *we have* $\nabla f(\mathbf{x}) = [3, -6, -1]$.

If $f \in C^2$ (partial second derivatives exist) then we define the *Hessian* of f at $\mathbf{x}$ to be the $n \times n$ matrix, denoted by $\nabla^2 f(\mathbf{x})$ or $\mathbf{H}(\mathbf{x})$ as

$$\mathbf{H}(\mathbf{x}) = \left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right]_{i,j=1}^{n}.$$

Since

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$$

it is obvious that the Hessian is symmetric. The $\dfrac{\partial^2 f}{\partial x_i \partial x_i}$ term is denoted by $\dfrac{\partial^2 f}{\partial x_i^2}$. Though the entries of the Hessian are functions, for any given $\mathbf{x}$ they determine a matrix of numerical values.

Example 49 *Let again* $f(\mathbf{x}) = (x_1 - 2x_2)^3 - 3x_3^3$. *Then*

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 6(x_1 - 2x_2) & -12(x_1 - 2x_2) & 0 \\ -12(x_1 - 2x_2) & 24(x_1 - 2x_2) & 0 \\ 0 & 0 & -18x_3 \end{bmatrix}.$$

At point $\mathbf{x} = [1, 1, -1/3]^T$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} -6 & 12 & 0 \\ 12 & -24 & 0 \\ 0 & 0 & 6 \end{bmatrix}.$$

The partial derivatives of an n-variable function $f(\mathbf{x})$ can characterize the extreme points of the function. It can be proved that for a point $\mathbf{x}_0$ to be an extreme point, a necessary condition is that the gradient in this point vanishes, i.e., $\nabla f(\mathbf{x}_0) = \mathbf{0}$. If, in this point, the Hessian exists and is positive definite this condition is sufficient and f has a minimum in $\mathbf{x}_0$. Similarly, if the Hessian exists and is negative definite, $\mathbf{x}_0$ is a maximizer.

As an interesting application of this observation, we develop the *normal equations* of the least squares problems.

Recall that the least squares problem was formulated as $\min \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$. This is equivalent to the minimization of the square of the 2-norm:

$$\min \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2.$$

If the square of the norm in an arbitrary point $\mathbf{x}$ is denoted by $f(\mathbf{x})$, i.e.,

$$f(\mathbf{x}) = (\mathbf{b} - \mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}),$$

then what we want to find is the minimum of $f(\mathbf{x})$. To this end, we must find a point where the partial derivatives vanish and verify that in this point the Hessian is positive definite. Writing out f, we obtain:

$$f(\mathbf{x}) = (\mathbf{b} - \mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}) = \sum_{i=1}^{m} \left[b_i - (a_{i1}x_1 + \cdots + a_{in}x_n)\right]^2 .$$

From this,

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \quad -2\sum_{i=1}^{m} \left[b_i - (a_{i1}x_1 + \cdots + a_{in}x_n)\right] a_{i1} &&= 0 \\ \vdots \quad & \qquad\qquad\qquad\qquad \vdots && \ \ \vdots \\ \frac{\partial f}{\partial x_n} &= \quad -2\sum_{i=1}^{m} \left[b_i - (a_{i1}x_1 + \cdots + a_{in}x_n)\right] a_{in} &&= 0 \end{aligned}$$

After simplification and rearrangement and remembering that $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n] = \begin{bmatrix} \mathbf{a}^1 \\ \mathbf{a}^2 \\ \vdots \\ \mathbf{a}^m \end{bmatrix}$ we obtain

$$\sum_{i=1}^{m} a_{i1} \overbrace{(a_{i1}x_1 + \cdots + a_{in}x_n)}^{\mathbf{a}^i\mathbf{x}} = \sum_{i=1}^{m} a_{i1}b_i \tag{72}$$

$$\vdots$$

$$\sum_{i=1}^{m} a_{in} \overbrace{(a_{i1}x_1 + \cdots + a_{in}x_n)}^{\mathbf{a}^i\mathbf{x}} = \sum_{i=1}^{m} a_{in}b_i \tag{73}$$

Recalling the definitions of vector and matrix operations, $(a_{i1}x_1 + \cdots + a_{in}x_n)$ in the sum on the left hand side is the dot product of row i of $\mathbf{A}$ and $\mathbf{x}$, i.e., $\mathbf{a}^i\mathbf{x}$. So, the left hand side of (72) is

$$\sum_{i=1}^{m} a_{i1}(\mathbf{a}^i\mathbf{x}) = a_{11}(\mathbf{a}^1\mathbf{x}) + a_{21}(\mathbf{a}^2\mathbf{x}) + \cdots + a_{m1}(\mathbf{a}^m\mathbf{x}) = \mathbf{a}_1^T(\mathbf{A}\mathbf{x}).$$

Obviously, the last row, i.e., the left hand side of (73), is $\mathbf{a}_n^T(\mathbf{A}\mathbf{x})$. Therefore, the sums in the rows are the dot products of the columns of $\mathbf{A}$ and the vector $\mathbf{A}\mathbf{x}$: $\mathbf{a}_1^T\mathbf{A}\mathbf{x}, \ldots, \mathbf{a}_n^T\mathbf{A}\mathbf{x}$. But they are the components of the vector $\mathbf{A}^T\mathbf{A}\mathbf{x}$. So, the left hand side is $\mathbf{A}^T\mathbf{A}\mathbf{x}$. The expressions on the right hand side are $\mathbf{a}_1^T\mathbf{b}, \ldots, \mathbf{a}_n^T\mathbf{b}$ which are the components of the $\mathbf{A}^T\mathbf{b}$ product. We conclude that an $\mathbf{x}$ to minimize the least squares must satisfy

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b},$$

which is the well known set of normal equations (see section 3.7).

We still need the positive definiteness of the Hessian. From the above, it can be seen that the Hessian is a constant matrix in the whole domain and is equal to $\mathbf{A}^T\mathbf{A}$ which is positive definite if the columns of $\mathbf{A}$ linearly independent. Therefore, the solution indeed minimizes function $f(\mathbf{x})$, the least squares.

5.2 Taylor Expansion

A group of important results of multivariate analysis are often referred to under the general heading of Taylor's Theorem. It enables us to approximate a function with an m-degree polynomial in a domain if it satisfies certain conditions.

First, we demonstrate the expansion for a function $f(x)$ with $m = 1$ (linear approximation). If f is twice differentiable in an $[x_0, x]$ interval then

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(\xi).$$

where f' and f'' denote the first and second derivatives, respectively, and ξ lies in the $[x_0, x]$ interval. The $\frac{1}{2}(x - x_0)^2 f''(\xi)$ part is the error term which is small if x is close to x_0 and the second derivative is bounded in the interval. From this formula, the Newton method for finding the root of an algebraic equation can be derived.

For a quadratic approximation we have to assume that f is three times differentiable. Details are not discussed here.

For a function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, the first order version of the theorem (linear approximation) assumes that $f \in C^1$ in a region containing the line segment $[\mathbf{x}_0, \mathbf{x}]$. It states that in this case there exists a θ, $0 \le \theta \le 1$, such that

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\theta \mathbf{x}_0 + (1 - \theta)\mathbf{x})(\mathbf{x} - \mathbf{x}_0).$$

Verbally, the function at $\mathbf{x}$ can be expressed as its value at $\mathbf{x}_0$ plus the gradient evaluated at some point in the line segment of $[\mathbf{x}_0, \mathbf{x}]$ (which is given as a convex linear combination of the two end points) times the difference of $\mathbf{x}$ and $\mathbf{x}_0$. This formula can be used to estimate the behaviour of the function in the neighbourhood of a given point ($\mathbf{x}_0$, in this case) when we have some information about the gradient.

If $f \in C^2$ then the second order version (quadratic approximation) says that there exists a θ, $0 \le \theta \le 1$, such that

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \\ &\quad \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \mathbf{H}(\theta \mathbf{x}_0 + (1 - \theta)\mathbf{x})(\mathbf{x} - \mathbf{x}_0), \end{aligned}$$

where $\mathbf{H}$ denotes the Hessian of f.

5.3 Newton's Method for $\min f(\mathbf{x})$

The idea behind Newton's method is that the function $f(\mathbf{x})$, $f \in C^2$, to be minimized is approximated locally by a quadratic function and this approximate function is minimized exactly. Thus, near a point $\mathbf{x}_k$ we can approximate $f(\mathbf{x})$ by the truncated Taylor series (omitting error term)

$$f(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T \mathbf{H}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k).$$

The necessary condition for a local minimum is that the gradient of $f(\mathbf{x})$ vanishes

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^T \mathbf{H}(\mathbf{x}_k) = \mathbf{0}^T,$$

which gives

$$\mathbf{x} = \mathbf{x}_k - \mathbf{H}^{-1}(\mathbf{x}_k)\left(\nabla f(\mathbf{x}_k)\right)^T.$$

This leads to the following iterative scheme:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}^{-1}(\mathbf{x}_k)\left(\nabla f(\mathbf{x}_k)\right)^T,$$

which is the pure form of Newton's method. If $\{\mathbf{x}_k\} \to \mathbf{x}^*$ and at $\mathbf{x}^*$ the Hessian $\mathbf{H}(\mathbf{x}^*)$ is positive definite then function $f(\mathbf{x})$ has a local minimum at $\mathbf{x}^*$. This method has an excellent (quadratic) convergence rate in the neighbourhood of a local minimum. To make it convergent in a wider radius, some more involved study is necessary.

5.4 Quadratic Forms and Linear Systems

In this section we show how the solution of $\mathbf{Ax} = \mathbf{b}$ can be associated with the minimization of a convex quadratic function. As a result, a very efficient iterative algorithm will be developed for solving $\mathbf{Ax} = \mathbf{b}$ in the case when $\mathbf{A}$ is positive definite. It will also be shown that, with a little trick, the method can be used to solve systems $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is not positive definite, not symmetric, and not even square.

The iterates will be denoted by $\mathbf{x}_k,\ k = 0, 1, 2, \dots,$ the corresponding residuals by $\mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$. Error is defined as $\mathbf{h}_k = \mathbf{x}_k - \mathbf{x}$, where $\mathbf{x}$ denotes the solution of $\mathbf{Ax} = \mathbf{b}$.

5.4.1 The quadratic form

A *quadratic form* is a real valued quadratic function of a vector $\mathbf{x}$:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Ax} - \mathbf{b}^T\mathbf{x} + c \tag{74}$$

where $\mathbf{A} \in \mathbb{R}^{m\times m}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^m$ and c is a scalar. We can try to minimize $f(\mathbf{x})$ by setting the gradient $f'(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{0}$. It can be shown that

$$(\nabla f(\mathbf{x}))^T = \frac{1}{2}\mathbf{A}^T\mathbf{x} + \frac{1}{2}\mathbf{Ax} - \mathbf{b}. \tag{75}$$

If $\mathbf{A}$ is symmetric, i.e., $\mathbf{A}^T = \mathbf{A}$, (75) reduces to

$$(\nabla f(\mathbf{x}))^T = \mathbf{Ax} - \mathbf{b}. \tag{76}$$

Setting (76) equal to $\mathbf{0}$ we obtain $\mathbf{Ax} = \mathbf{b}$. It is easy to see that the constant matrix $\mathbf{A}$ is the Hessian of $f(\mathbf{x})$ for any $\mathbf{x}$. Therefore, if it is positive definite then the solution of $\mathbf{Ax} = \mathbf{b}$ minimizes $f(\mathbf{x})$ in the global sense. Conversely, if $\mathbf{x}$ minimizes $f(\mathbf{x})$ then it also solves $\mathbf{Ax} = \mathbf{b}$.

Since $\nabla f(\mathbf{x})$ points to the direction of greatest increase of $f(\mathbf{x})$, its negation $-(\nabla f(\mathbf{x}))^T = \mathbf{b} - \mathbf{Ax}$ is the direction of the greatest decrease of $f(\mathbf{x})$.

5.4.2 Iterative methods with search direction

A large family of *iterative methods* for minimizing a function $f(\mathbf{x})$ work in the following way. Identify a *search direction*, say $\mathbf{d}$, along which f starts to descend and move from the current iterate into this direction. The *steplength* of the movement, say α, is usually determined such that it gives the largest possible improvement of the value of $f(\mathbf{x})$ in this direction. So, the following sequence is generated:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k.$$

Therefore, these methods involve two main steps per iteration: (i) finding a search direction and (ii) determining the steplength.

5.4.3 The Steepest Descent Method (SDM)

The Steepest Descent Method (SDM) is an iterative method with search direction for finding the minimum of quadratic form (74). It also solves $\mathbf{Ax} = \mathbf{b}$ if $\mathbf{A}$ is positive definite.

In each iteration k, SDM chooses the direction in which f decreases most quickly which is the negative of the gradient: $-(\nabla f(\mathbf{x}_k))^T = \mathbf{b} - \mathbf{Ax}_k$. It is nothing but the residual $\mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$. It means that the search direction is the residual vector. Thus the next iterate is computed as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{r}_k.$$

In this direction f is a function of α_k

$$f(\mathbf{x}_k + \alpha_k\mathbf{r}_k)$$

We want to determine the value of α_k that minimizes f (*line search*). It can be achieved by setting the *directional derivative* $\dfrac{df(\mathbf{x}_{k+1})}{d\alpha_k}$ equal to zero. By the chain rule we obtain

$$\frac{df(\mathbf{x}_{k+1})}{d\alpha_k} = f'(\mathbf{x}_{k+1})\frac{d\mathbf{x}_{k+1}}{d\alpha_k} = \nabla f(\mathbf{x}_{k+1})\mathbf{r}_k = -\mathbf{r}_{k+1}^T\mathbf{r}_k = 0.$$

The latter suggests that α_k must be determined such that $\mathbf{r}_{k+1}$ becomes orthogonal to $\mathbf{r}_k$. This observation enables us to find this value as a result of the following simple development:

$$\begin{aligned}
\mathbf{r}_{k+1}^T\mathbf{r}_k &= 0\\
(\mathbf{b}-\mathbf{A}\mathbf{x}_{k+1})^T\mathbf{r}_k &= 0\\
(\mathbf{b}-\mathbf{A}(\mathbf{x}_k+\alpha_k\mathbf{r}_k))^T\mathbf{r}_k &= 0\\
(\mathbf{b}-\mathbf{A}\mathbf{x}_k)^T\mathbf{r}_k - \alpha_k(\mathbf{A}\mathbf{r}_k)^T\mathbf{r}_k &= 0\\
\alpha_k(\mathbf{A}\mathbf{r}_k)^T\mathbf{r}_k &= (\mathbf{b}-\mathbf{A}\mathbf{x}_k)^T\mathbf{r}_k\\
\alpha_k(\mathbf{A}\mathbf{r}_k)^T\mathbf{r}_k &= \mathbf{r}_k^T\mathbf{r}_k\\
\alpha_k &= \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}.
\end{aligned}$$

Now we are ready to state the first version of the Steepest Descent algorithm:

$$\mathbf{r}_k = \mathbf{b}-\mathbf{A}\mathbf{x}_k \tag{77}$$

$$\alpha_k = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k} \tag{78}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k+\alpha_k\mathbf{r}_k. \tag{79}$$

These operations are performed in each iteration. Iterations are repeated until $\mathbf{x}_k$ converges. The algorithm in its current form requires two matrix-vector multiplications per iteration. With a little trick, one of them can be eliminated. Namely, by premultiplying both sides of (79) by $-\mathbf{A}$ and adding $\mathbf{b}$, we obtain:

$$\begin{aligned}
\mathbf{b}-\mathbf{A}\mathbf{x}_{k+1} &= (\mathbf{b}-\mathbf{A}\mathbf{x}_k)-\alpha_k\mathbf{A}\mathbf{r}_k\\
\mathbf{r}_{k+1} &= \mathbf{r}_k-\alpha_k\mathbf{A}\mathbf{r}_k. \qquad (80)
\end{aligned}$$

In this way, (77) is needed only for $k=0$ to start the algorithm. In subsequent iterations (80) can be used instead. The product $\mathbf{A}\mathbf{r}_k$ which occurs in both (78) and (80) has to be computed only once per iteration. This observation is generally helpful because Steepest Descent is dominated by matrix-vector products. However, as floating point computational errors accumulate it is worthwhile to periodically recompute the residual from its definition $\mathbf{r}_k=\mathbf{b}-\mathbf{A}\mathbf{x}_k$.

Below, we give a step-by-step description of the Steepest Descent Method for solving $\mathbf{A}\mathbf{x}=\mathbf{b}$ if $\mathbf{A}$ is an $m\times m$ positive definite matrix. In the description, for simpler presentation, we omit the iteration count k and use ":=" as an assignment operator.

Inputs to the algorithm are: $\mathbf{A}$, $\mathbf{b}$, an initial guess of the solution $\mathbf{x}$, the maximum number of iterations $k_{\max}$ and an error tolerance $0<\epsilon<1$.

```
$k := 0$
$\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$
$\delta := \mathbf{r}^T\mathbf{r}$
$\delta_0 := \delta$
While $k < k_{\max}$ and $\delta > \epsilon^2\delta_0$ do
    $\mathbf{q} := \mathbf{A}\mathbf{r}$
    $\alpha := \frac{\delta}{\mathbf{r}^T\mathbf{q}}$
    $\mathbf{x} := \mathbf{x} + \alpha\mathbf{r}$
    If $k$ is divisible by 50 then
        $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$
    else
        $\mathbf{r} := \mathbf{r} - \alpha\mathbf{q}$
    end if
    $\delta := \mathbf{r}^T\mathbf{r}$
    $k := k + 1$
end do
```

The algorithm terminates when the maximum number of iterations $k_{\max}$ has been reached, or sooner if in iteration k we observe that $||\mathbf{r}_k||_2^2 \leq \epsilon^2||\mathbf{r}_0||_2^2$.

The residual is updated using the fast formula in (80). However, once in every 50 iterations the exact residual is recalculated to get rid of accumulated floating point error. This recomputation frequency can also be an input parameter to the algorithm. If the tolerance ϵ is close to the relative accuracy of the floating point precision of the processor, a test should be added after δ is evaluated to check if $\delta \leq \epsilon^2\delta_0$, and if this test holds true, the exact residual should also be recomputed and δ reevaluated. This prevents procedure from terminating early due to floating point round-off error.

This algorithm is generally good if the spectral radius $\rho(\mathbf{A}) < 1$. However, in some cases it can take a zigzagging path with small steps. The reason for it is that each search direction is orthogonal to the previous one and search directions can repeat.

5.4.4 The Method of Conjugate Directions (MCD)

To get around the problem of repeating search directions one could use predetermined directions that are orthogonal to all earlier ones. While the coordinate axes are natural candidates for this they do not lead to computationally useful formulae. However, there is a remedy.

Conjugacy is a generalization of the idea of orthogonality of two m-vectors. Orthogonality of $\mathbf{u}$ and $\mathbf{v}$ was defined as $\mathbf{u}^T\mathbf{v} = 0$. By inserting the identity matrix: $\mathbf{u}^T\mathbf{I}\mathbf{v} = 0$. If $\mathbf{I}$ is replaced by an $m \times m$ matrix $\mathbf{A}$ then we say that the two vectors, $\mathbf{u}$ and $\mathbf{v}$, are A-orthogonal or *conjugate* (with respect to $\mathbf{A}$) if

$$\mathbf{u}^T\mathbf{A}\mathbf{v} = 0.$$

Example 50 *Let*

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \quad \textit{and} \quad \mathbf{u}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \; \mathbf{v}_1 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \quad \textit{then} \quad \mathbf{u}_1^T\mathbf{A}\mathbf{v}_1 = 0,$$

i.e., $\mathbf{u}_1$ *and* $\mathbf{v}_1$ *are A-orthogonal (but not orthogonal). Similarly,* $\mathbf{u}_2 = [2, -3]^T$ *and* $\mathbf{v}_2 = [1, 0]^T$ *are also A-orthogonal with the same* $\mathbf{A}$ *but* $\mathbf{u}_3 = [1, 0]^T$ *and* $\mathbf{v}_3 = [0, 1]^T$ *are not.*

It can be shown that if $\mathbf{A}$ is positive definite then there are m A-orthogonal (conjugate) vectors $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{m-1}$ so that $\mathbf{d}_i^T\mathbf{A}\mathbf{d}_j = 0$ for all $0 \leq i, j \leq m-1,\ i \neq j$. They are actually directions because any nonzero scalar multiples of the vectors are also conjugate.

Using these directions, we still need to perform line search to determine the steplength α_k. This is achieved by setting the directional derivative equal to 0 as in the case of Steepest Descent. Omitting the similar details, finally we obtain

$$\alpha_k = \frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}$$

Note that if the search vector were the residual, this formula would be identical to the one used in SDM.

Now the problem is 'reduced' to finding m conjugate directions. This, in itself, would require a procedure equivalent to the Gaussian elimination. However, taking advantage of the properties of conjugacy, there is a way (details not presented here) to determine the directions 'on-the-fly' using the Gram-Schmidt conjugation procedure. This results in the first version of the Method of Conjugate Directions:

$$\begin{aligned}
\mathbf{d}_0 = \mathbf{r}_0 &= \mathbf{b} - \mathbf{A}\mathbf{x}_0 && (81)\\
\alpha_k &= \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k} && (82)\\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{d}_k && (83)\\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{d}_k && (84)\\
\beta_{k+1} &= \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} && (85)\\
\mathbf{d}_{k+1} &= \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{d}_k. && (86)
\end{aligned}$$

This algorithm is also known as *conjugate gradient (CG) method* though this name is slightly misleading. It finds the minimum of $f(\mathbf{x})$ which will, at the same time, solve $\mathbf{A}\mathbf{x} = \mathbf{b}$.

After some refinements, the Method of Conjugate Directions can be stated in the following way. The algorithm can be used to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ if $\mathbf{A}$ is an $m \times m$ positive definite matrix. Again, in the description ":=" is used as an assignment operator while the iteration index k is omitted.

Inputs to the algorithm are: $\mathbf{A}$, $\mathbf{b}$, an initial guess of the solution $\mathbf{x}$, the maximum number of iterations $k_{\max}$ and an error tolerance $0 < \epsilon < 1$.

```
$k := 0$
$\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$
$\mathbf{d} := \mathbf{r}$
$\delta_{\text{new}} := \mathbf{r}^T\mathbf{r}$
$\delta_0 := \delta_{\text{new}}$
**While** $k < k_{\max}$ **and** $\delta_{\text{new}} > \epsilon^2 \delta_0$ **do**
    $\mathbf{q} := \mathbf{A}\mathbf{d}$
    $\alpha := \frac{\delta_{\text{new}}}{\mathbf{d}^T\mathbf{q}}$
    $\mathbf{x} := \mathbf{x} + \alpha\mathbf{d}$
    **If** $k$ is divisible by 50 **then**
        $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$
    **else**
        $\mathbf{r} := \mathbf{r} - \alpha\mathbf{q}$
    **end if**
    $\delta_{\text{old}} := \delta_{\text{new}}$
    $\delta_{\text{new}} := \mathbf{r}^T\mathbf{r}$
    $\beta := \frac{\delta_{\text{new}}}{\delta_{\text{old}}}$
    $\mathbf{d} := \mathbf{r} + \beta\mathbf{d}$
    $k := k + 1$
**end do**
```

The same comments apply to this algorithm as given at the end of the Steepest Descent Method.

It can be shown that MCD finds the minimum of $f(\mathbf{x})$ defined in (74) and, therefore, solves $\mathbf{Ax} = \mathbf{b}$ in not more than m iterations if $\mathbf{A}$ is positive definite. In such a case it is not a real iterative method since the number of steps is known in advance.

5.4.5 MCD for general $\mathbf{Ax} = \mathbf{b}$

In this section, for notational convenience, the system of linear equations will be represented as $\mathbf{Gx} = \mathbf{g}$. If matrix $\mathbf{G}$ is not symmetric, not positive definite or not even square we can attempt to find an $\mathbf{x}$ vector that minimizes

$$f(\mathbf{x}) = \|\mathbf{Gx} - \mathbf{g}\|_2^2,$$

which is the already known least squares problem. If at the minimum $f(\mathbf{x}) = 0$ we have a solution to $\mathbf{Gx} = \mathbf{g}$. Otherwise, we have a solution that satisfies the equations with the least error (in ℓ_2 sense). The minimum of $f(\mathbf{x})$ is attained where the gradient vanishes: $\nabla f(\mathbf{x}) = 0$, from which we obtain the normal equations:

$$\mathbf{G}^T\mathbf{Gx} = \mathbf{G}^T\mathbf{g}.$$

Now, we have a system of linear equations with positive definite matrix $\mathbf{G}^T\mathbf{G}$ (assuming linear independence of the columns of $\mathbf{G}$ which is very likely the case for overdetermined systems: $m > n$). As such, MCD can be applied to it by substituting $\mathbf{G}^T\mathbf{G}$ for $\mathbf{A}$ and $\mathbf{G}^T\mathbf{g}$ for $\mathbf{b}$. The only concern in using this method is that the condition number of $\mathbf{G}^T\mathbf{G}$ is the square of that of $\mathbf{G}$ and if it is a large number then convergence can be slow.

There is an important computational aspect to be considered. $\mathbf{G}^T\mathbf{G}$ is never computed explicitly. The reason is that $\mathbf{G}$ is usually very sparse in case of large scale problems. However, if $\mathbf{G}^T\mathbf{G}$ is formed it can be completely dense rendering the method computationally unattractive. The remedy is to interpret $\mathbf{Ad}_k$ of (84) as $\mathbf{G}^T\mathbf{Gd}_k = \mathbf{G}^T(\mathbf{Gd}_k)$ and first compute $\mathbf{Gd}_k$ (sparse matrix-vector product) then multiply the result by $\mathbf{G}^T$ (sparse product, again). Similarly, for the same reasons, $\mathbf{d}_k^T\mathbf{Ad}_k$ in (82) can be computed as $(\mathbf{d}_k^T\mathbf{G}^T)(\mathbf{Gd}_k)$.

5.4.6 MCD for arbitrary nonlinear $f(\mathbf{x})$

The Method of Conjugate Directions can be used not only to find the minimum point of a quadratic form but also to minimize any continuous function $f(\mathbf{x})$ for which the gradient $f'(\mathbf{x}) = \nabla f(\mathbf{x})$ can be computed. The principles are similar to the quadratic case but the derivation is more complicated and is beyond the scope of this course. It is still important to note that the nonlinear MCD is the method of choice in many applications where optimization problems are to be solved, such as engineering design, neural net training, and nonlinear regression.

Index