

Objectives:

- Review matrix notation.
- Systems of equations.
- Gaussian elimination.
- **Materials:** Hornberger and Wiberg, Chpt. 2; Forsythe et al., Chpt. 3; Gerald and Wheatley, Chpt. 2.

Matrix notation

A matrix is simply a two-dimensional array of numbers, say of size $m \times n$. In this notation, the first dimension, “m” is the number of rows, and the second dimension is the number of columns. The elements of a matrix, A , are often designated to be a_{ij} , where $i = 1, 2, \dots, m$, and $j = 1, 2, \dots, n$. A scalar is a special case that is a 1×1 number (0 dimensional). A vector is a special case, $n \times 1$ or $1 \times n$, of a one-dimensional array. A *column vector* (dimensions $n \times 1$) commonly occurs. Multi-dimensional arrays are often designated using a capital letter (A or B , etc.), and scalars and vectors using little letters (x or b , etc.).

Matrix algebra is similar to, but different from, scalar algebra. Matrix addition and subtraction ($A + B$) are defined if both A and B are the same size. The sum $A + B = C$ is defined by $c_{ij} = a_{ij} + b_{ij}$. Matrix multiplication is a little more complicated. The operation $C = A B$ is defined if A is size $m \times n$, and B is size $n \times p$, i.e. if B has the same number of rows as A has columns.

$$\begin{aligned} C &= AB \\ c_{ij} &= a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj} \\ c_{ij} &= \sum_{k=1, \dots, n} a_{ik}b_{kj}; \end{aligned} \tag{1}$$

Note that matrix multiplication is not commutative; that is $AB \neq BA$. A special case of vector multiplication of an $1 \times n$ vector, x , to an $n \times 1$ vector, y is the *scalar product* or *inner product*,

$$\begin{aligned} z &= xy; \\ z &= \sum_{k=1, \dots, n} x_{1k}y_{k1} \end{aligned} \tag{2}$$

The **transpose** of a matrix, $B = A^T$ (or in matlab $B = A'$) is the matrix B , written so that the rows of B equal the columns of A . That is $b_{ij} = a_{ji}$.

A matrix, A , is defined to be an **upper triangular** matrix if all of the elements below the diagonal are zero; i.e. $a_{ij} = 0$ for $i > j$. It is defined to be an **lower triangular** matrix if all of the elements above the diagonal are zero; i.e. $a_{ij} = 0$ for $i < j$. Upper triangular, and lower triangular matrices are often designated with the symbols U and L , respectively.

Sparse matrices are matrices in which the majority of the elements are zero. These come up frequently in numerical methods, and specialized, efficient techniques have been derived to deal with them.

A square matrix, A , is defined to be **diagonal** if the only non-zero elements are those along the main diagonal, that is $a_{ij} = 0$ for $i \neq j$. A matrix, A , is defined to be **banded** if it only has non-zero elements on the diagonal, and on the off-diagonals close to the main diagonal. A **banded matrix** of rank $2m + 1$ has non-zero elements only within “ m ” sub-diagonals of the main-diagonal. That is, $a_{ij} = 0$ for $|i - j| > m$. A common occurrence in the types of finite difference methods that scientists use is to have a special case of a banded matrix; a **tri-diagonal** matrix A , where the elements of A are zero for $a_{ij} = 0$ for $|i - j| > 1$.

Another important special case of a diagonal matrix is the **identity matrix**, I . This matrix is a diagonal matrix with all of the diagonal elements equal to 1. The $n \times n$ identity matrix, I has the property that $Ix = xI = x$, and $IA = AI = A$ for an $n \times 1$ vector x , or an $n \times n$ matrix A . It would be called an **identity matrix of order n** , and designated I_n . If you exchange some of the rows of an identity matrix, you get a **transportation matrix**, P , that is sometimes useful.

The **inverse** of a square matrix, A , is another important concept. The inverse, designated A^{-1} is the matrix such that $A^{-1}A = AA^{-1} = I$. Not all square matrices have an inverse, and computing the inverse of a square matrix can be a computationally expensive enterprise. Matlab has a function, `inv()` that will estimate an inverse.

The **eigenvector**, w , and associated **eigenvalue**, λ , of the matrix A are defined: $Aw = \lambda w$. This can also be written $(A - \lambda I)w = 0$.

Systems of linear equations

A common problem in scientific computing is the need to solve a system of “ n ” linear equations of “ n ” variables:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1; \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2; \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n; \end{aligned} \tag{3}$$

where the a_{ij} , and b_j are known, and the x_i are unknown. This is called a **system of linear equations**, and it can be rewritten as the matrix equation $Ax = b$. A will be a square, $n \times n$ matrix, and b and x are vectors of size $n \times 1$.

One way to solve the system is to find the inverse of A , A^{-1} ; since $A^{-1}Ax = A^{-1}b$; $Ix = A^{-1}b$, and $x = A^{-1}b$. So, if you can find $A^{-1} = inv(A)$, you can multiply it by the vector “b” and get your solution. Unfortunately, finding A^{-1} is, in general, computationally expensive.

The matrix A is defined to be *singular* if no solution exists. This will be the case, for example, if any row of A is a linear combination of other rows of A . In that case, the matrix A will not have an inverse.

Gaussian elimination

This technique is commonly used to solve systems of linear equations, $Ax = b$. In many circumstances, it is much more efficient than attempting to find the inverse of the matrix, A . The technique relies on a series of linear operations that act to transform the matrix, A , and right-hand-side, b , so that A is transformed into an upper-triangular matrix. Forsythe, et al. summarize the steps. The main thing to remember is that the system, $Ax = b$ is a system of linear equations, $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_n$, so that linear transformations work on the matrices the same way that they’d work on the equations. That is- you can multiply both the right-hand-side (b), and the left-hand side (Ax) by a scalar, and you can add rows of the system to each other. These linear operations are done on the matrix until it is transformed to $Ux = b'$, where U is an upper triangular matrix with 1’s along the diagonal. The values of x are then easily solved.

$$\begin{aligned}
 Ux = b' &= \\
 x_1 + u_{12}x_2 + \dots + u_{1n}x_n &= b'_1; \\
 &\vdots \\
 x_{n-1} + u_{n-1,n}x_n &= b'_{n-1} \\
 x_n &= b'_n;
 \end{aligned} \tag{4}$$