MATRICES

(slightly modified content from Wikipedia articles on matrices http://en.wikipedia.org/wiki/Matrix_(mathematics))

A *matrix* is a rectangular array of numbers or other mathematical objects, for which operations such as addition and multiplication are defined. Most of this article focuses on *real matrices*, i.e., matrices whose elements are real numbers. For instance, this is a real matrix:

$$\mathbf{A} = \begin{bmatrix} -1.3 & 0.6 \\ 20.4 & 5.5 \\ 9.7 & -6.2 \end{bmatrix}.$$

The numbers, symbols or expressions in the matrix are called its *entries* or its *elements*. The horizontal and vertical lines of entries in a matrix are called *rows* and *columns*, respectively. The size of a matrix is defined by the number of rows and columns that it contains. A matrix with *m* rows and *n* columns is called an *m* × *n* matrix or *m*-by-*n* matrix, while *m* and *n* are called its *dimensions*. For example, the matrix **A** above is a 3 × 2 matrix. Matrices which have a single row are called *row vectors*, and those which have a single column are called *column vectors*. A matrix which has the same number of rows and columns is called a *square matrix*.

Name	Size	Example	Description
Row vector	$1 \times n$	$\begin{bmatrix} 3 & 7 & 2 \end{bmatrix}$	A matrix with one row, sometimes used to represent a vector
Column vector	$n \times 1$	$\begin{bmatrix} 4 \\ 1 \\ 8 \end{bmatrix}$	A matrix with one column, sometimes used to represent a vector
Square matrix	$n \times n$	$\begin{bmatrix} 9 & 13 & 5 \\ 1 & 11 & 7 \\ 2 & 6 & 3 \end{bmatrix}$	A matrix with the same number of rows and columns.

1. Notation

Matrices are commonly written in box brackets:

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

The specifics of symbolic matrix notation varies widely, with some prevailing trends. Matrices are usually symbolized using upper-case letters (such as **A** in the examples above), while the corresponding lower-case letters, with two subscript indices (e.g., a_{11}, or $a_{1,1}$), represent the entries.

The entry in the i-th row and j-th column of a matrix **A** is sometimes referred to as the i,j, (i,j), or $(i,j)^{\text{th}}$ entry of the matrix, and most commonly denoted as $a_{i,j}$, or a_{ij}. Alternative notations for that entry are $A[i,j]$ or $A_{i,j}$. For example, the (1,3) entry of the following matrix **A** is 5 (also denoted a_{13}, $a_{1,3}$, $A[1,3]$ or $A_{1,3}$):

$$\mathbf{A} = \begin{bmatrix} 4 & -7 & 5 & 0 \\ -2 & 0 & 11 & 8 \\ 19 & 1 & -3 & 12 \end{bmatrix}$$

Sometimes, the entries of a matrix can be defined by a formula such as $a_{i,j} = f(i, j)$. For example, each of the entries of the following matrix **A** is determined by $a_{ij} = i - j$.

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

In this case, the matrix itself is sometimes defined by that formula, within square brackets or double parenthesis. For example, the matrix above is defined as $\mathbf{A} = [i\text{-}j]$. If matrix size is $m \times n$, the above-mentioned formula $f(i, j)$ is valid for any $i = 1, ..., m$ and any $j = 1, ..., n$. This can be either specified separately, or using $m \times n$ as a subscript. For instance, the matrix **A** above is 3×4 and can be defined as $\mathbf{A} = [i - j]$ $(i = 1, 2, 3; j = 1, ..., 4)$, or $\mathbf{A} = [i - j]_{3\times4}$.

Submatrix

A **submatrix** of a matrix is obtained by deleting any collection of rows and/or columns. For example, from the following 3-by-4 matrix, we can construct a 2-by-3 submatrix by removing row 3 and column 2:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 3 & 4 \\ 5 & 7 & 8 \end{bmatrix}.$$

A **principal submatrix** is a square submatrix obtained by removing certain rows and columns. A leading principal submatrix is one in which the first k rows and columns, for some number k, are the ones that remain.

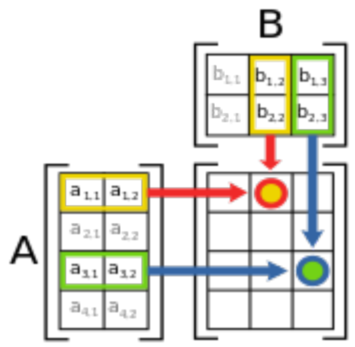
2. Basic operations

There are a number of basic operations that can be applied to modify matrices, called *matrix addition*, *scalar multiplication*, *transposition*, *matrix multiplication*, *row operations*, and *submatrix*.

Operation	Definition	Example
Addition	The *sum* **A**+**B** of two *m*-by-*n* matrices **A** and **B** is calculated entrywise: $(\mathbf{A}+\mathbf{B})_{i,j} = \mathbf{A}_{i,j} + \mathbf{B}_{i,j}$, where $1 \le i \le m$ and $1 \le j \le n$.	$\begin{bmatrix}1 & 3 & 1\\1 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 5\\7 & 5 & 0\end{bmatrix} = \begin{bmatrix}1+0 & 3+0 & 1+5\\1+7 & 0+5 & 0+0\end{bmatrix} = \begin{bmatrix}1 & 3 & 6\\8 & 5 & 0\end{bmatrix}$
Scalar multiplication	The product *c***A** of a number *c* (also called a scalar) and a matrix **A** is computed by multiplying every entry of **A** by *c*: $(c\mathbf{A})_{i,j} = c \cdot \mathbf{A}_{i,j}$. This operation is called *scalar multiplication*.	$2 \cdot \begin{bmatrix}1 & 8 & -3\\4 & -2 & 5\end{bmatrix} = \begin{bmatrix}2\cdot 1 & 2\cdot 8 & 2\cdot -3\\2\cdot 4 & 2\cdot -2 & 2\cdot 5\end{bmatrix} = \begin{bmatrix}2 & 16 & -6\\8 & -4 & 10\end{bmatrix}$
Transposition	The *transpose* of an *m*-by-*n* matrix **A** is the *n*-by-*m* matrix $\mathbf{A}^T$ formed by turning rows into columns and vice versa: $(\mathbf{A}^T)_{i,j} = \mathbf{A}_{j,i}$.	$\begin{bmatrix}1 & 2 & 3\\0 & -6 & 7\end{bmatrix}^T = \begin{bmatrix}1 & 0\\2 & -6\\3 & 7\end{bmatrix}$

Familiar properties of numbers extend to these operations of matrices: for example, addition is commutative, i.e., the matrix sum does not depend on the order of the summands: $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$. The transpose is compatible with addition and scalar multiplication, as expressed by $(c\mathbf{A})^T = c(\mathbf{A}^T)$ and $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$. Finally, $(\mathbf{A}^T)^T = \mathbf{A}$.

Matrix multiplication



Schematic depiction of the matrix product **AB** of two matrices **A** and **B**.

Multiplication of two matrices is defined if and only if the number of columns of the left matrix is the same as the number of rows of the right matrix. If **A** is an m-by-n matrix and **B** is an n-by-p matrix, then their *matrix product* **AB** is the m-by-p matrix whose entries are given by dot product of the corresponding row of **A** and the corresponding column of **B**:

$$[\mathbf{AB}]_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \cdots + A_{i,n}B_{n,j} = \sum_{r=1}^{n} A_{i,r}B_{r,j},$$

where $1 \leq i \leq m$ and $1 \leq j \leq p$.[12] For example, the underlined entry 2340 in the product is calculated as $(2 \times 1000) + (3 \times 100) + (4 \times 10) = 2340$:

$$\begin{bmatrix} \underline{2} & \underline{3} & \underline{4} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \underline{1000} \\ 1 & \underline{100} \\ 0 & \underline{10} \end{bmatrix} = \begin{bmatrix} 3 & \underline{2340} \\ 0 & 1000 \end{bmatrix}.$$

Matrix multiplication satisfies the rules **(AB)C = A(BC)** (associativity), and **(A+B)C = AC+BC** as well as **C(A+B) = CA+CB** (left and right distributivity), whenever the size of the matrices is such that the various products are defined.[13] The product **AB** may be defined without **BA** being defined, namely if **A** and **B** are m-by-n and n-by-k matrices, respectively, and $m \neq k$. Even if both products are defined, they need not be equal, i.e., generally

AB ≠ BA,

i.e., *matrix multiplication is not commutative*, in marked contrast to (rational, real, or complex) numbers whose product is independent of the order of the factors. An example of two matrices not commuting with each other is:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 3 \end{bmatrix}, \text{ whereas } \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 0 & 0 \end{bmatrix}.$$

The following property for transpose of the matrix product can be shown to hold: $(\mathbf{AB})^{\mathbf{T}} = \mathbf{B}^{\mathbf{T}}\mathbf{A}^{\mathbf{T}}$

Besides the ordinary matrix multiplication just described, there exist other less frequently used operations on matrices that can be considered forms of multiplication, such as the Hadamard product and the Kronecker product.[14]

Row and Column operations

There are three types of row operations on a matrix **A** of dimension m-by-n that can be produced by multiplying it from left with an elementary matrix **E** of dimension m-by-m, $E{\cdot}A$:

1. row switching, that is interchanging two rows of a matrix.

2. row addition, that is adding a row to another.
3. row multiplication, that is multiplying all entries of a row by a non-zero constant.

In mathematics, an **elementary matrix** is a matrix which differs from the identity matrix by one single elementary row operation. Repeated multiplication of the identity matrix by the elementary matrices can generate any invertible matrix (definition of the inverse matrix will come later). Left multiplication (pre-multiplication) by an elementary matrix represents **elementary row operations**, while right multiplication (post-multiplication) represents **elementary column operations**. Elementary row operations are the basis of the very important method of Gaussian elimination (will be explained below).

Row-switching transformations

The first type of row operation on a matrix **A** switches all matrix elements on row *i* with their counterparts on row *j*. The corresponding elementary matrix is obtained by swapping row *i* and row *j* of the identity matrix.

$$T_{i,j} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

So $\mathbf{T}_{ij}\cdot\mathbf{A}$ is the matrix produced by exchanging row *i* and row *j* of **A**.

Row-multiplying transformations

The next type of row operation on a matrix **A** multiplies all elements on row *i* by *m* where *m* is a non-zero scalar (usually a real number). The corresponding elementary matrix is a diagonal matrix, with diagonal entries 1 everywhere except in the *i*th position, where it is *m*.

$$T_i(m) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & m & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

So $\mathbf{T}_i(m)\cdot\mathbf{A}$ is the matrix produced from **A** by multiplying row *i* by *m*.

Row-addition transformations

The final type of row operation on a matrix **A** adds row *j* multiplied by a scalar *m* to row *i*. The corresponding elementary matrix is the identity matrix but with an *m* in the (*i*,*j*) position.

$$T_{i,j}(m) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & m & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

So $\mathbf{T}_{i,j}(m)\cdot\mathbf{A}$ is the matrix produced from **A** by adding *m* times row *j* to row *i*.

3. Linear equations

Matrices can be used to compactly write and work with systems of linear equations. For example, if **A** is an *m*-by-*n* matrix, **x** designates a column vector (i.e., $n\times 1$-matrix) of *n* variables $x_1, x_2, \ldots, x_n$, and **b** is an $m\times 1$-column vector, then the matrix equation

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

is equivalent to the system of linear equations

$$a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1$$

$$\ldots$$

$$a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n = b_m \,.$$

For example,

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 1 \\ 2x_1 - 2x_2 + 4x_3 &= -2 \\ -x_1 + 1/2x_2 - x_3 &= 0 \end{aligned}$$

is a system of three equations in the three variables x_1, x_2, and x_3. This can be written in the matrix form $\mathbf{Ax} = \mathbf{b}$ where

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & 1/2 & -1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}$$

A **solution** to a linear system is an assignment of numbers to the variables such that all the equations are simultaneously satisfied. A solution to the system above is given by

$$\begin{aligned} x_1 &= 1 \\ x_2 &= -2 \\ x_3 &= -2 \end{aligned}$$

since it makes all three equations valid.

A linear system may behave in any one of three possible ways:

1. The system has *infinitely many solutions*.
2. The system has a single *unique solution*.
3. The system has *no solution*.

3.1. Solving linear equations

There are several algorithms for solving a system of linear equations.

Elimination of variables

The simplest method for solving a system of linear equations is to repeatedly eliminate variables. This method can be described as follows:

1. In the first equation, solve for one of the variables in terms of the others.
2. Substitute this expression into the remaining equations. This yields a system of equations with one fewer equation and one fewer unknown.
3. Continue until you have reduced the system to a single linear equation.
4. Solve this equation, and then back-substitute until the entire solution is found.

For example, consider the following system:

$$\begin{aligned} x + 3y - 2z &= 5 \\ 3x + 5y + 6z &= 7 \\ 2x + 4y + 3z &= 8 \end{aligned}$$

Solving the first equation for x gives $x = 5 + 2z - 3y$, and plugging this into the second and third equation yields

$$\begin{aligned} -4y + 12z &= -8 \\ -2y + 7z &= -2 \end{aligned}$$

Solving the first of these equations for y yields $y = 2 + 3z$, and plugging this into the second equation yields $z = 2$. We now have:

$$\begin{aligned} x &= 5 + 2z - 3y \\ y &= 2 + 3z \\ z &= 2 \end{aligned}$$

Substituting $z = 2$ into the second equation gives $y = 8$, and substituting $z = 2$ and $y = 8$ into the first equation yields $x = -15$. Therefore, the solution set is the single point $(x, y, z) = (-15, 8, 2)$.

Gaussian elimination

Elimination of variables method is natural for solving by hand, but is a little cumbersome to implement it as a computer program. The method of Gaussian elimination can be described as an easy to implement algorithm and is the standard algorithm for numerically solving a system of linear equations. Gaussian elimination relies on row-reduction of an augmented matrix as explained below.

Augmented matrix

Given a system of linear equation represented in the matrix form as $\mathbf{Ax} = \mathbf{b}$, its augmented matrix is obtained by concatenating m-by-1 column vector $\mathbf{b}$ to m-by-n $\mathbf{A}$ and creating m-by-(n+1) augmented matrix [$\mathbf{A}$ | $\mathbf{b}$].

For example,

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & 1/2 & -1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix} \quad \Rightarrow \quad [\mathbf{A} \mid \mathbf{b}] = \left[\begin{array}{ccc|c} 3 & 2 & -1 & 1 \\ 2 & -2 & 4 & -2 \\ -1 & 1/2 & -1 & 0 \end{array}\right]$$

Row reduction

In **row reduction**, the linear system is represented as an augmented matrix and this matrix is then modified using elementary row operations until it reaches reduced row echelon form. A matrix is in **row echelon form** if

- All nonzero rows (rows with at least one nonzero element) are above any rows of all zeroes (all zero rows, if any, belong at the bottom of the matrix).

- The leading coefficient (the first nonzero number from the left, also called the pivot) of a nonzero row is always strictly to the right of the leading coefficient of the row above it.

This is an example of a 3×5 matrix in row echelon form:

$$\begin{bmatrix} 1 & a_0 & a_1 & a_2 & a_3 \\ 0 & 0 & 2 & a_4 & a_5 \\ 0 & 0 & 0 & 1 & a_6 \end{bmatrix}$$

A matrix is in **reduced row echelon form** (also called **row canonical form**) if it satisfies the following conditions:

- It is in row echelon form.
- Every leading coefficient is 1 and is the only nonzero entry in its column.

This is an example of a matrix in reduced row echelon form:

$$\begin{bmatrix} 1 & 0 & a_1 & 0 & b_1 \\ 0 & 1 & a_2 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \end{bmatrix}$$

Because these operations are reversible, the augmented matrix produced always represents a linear system that is equivalent to the original.

The Algorithm

Gaussian elimination relies on the following 3 observations: (1) multiplying any equation by a scalar does not change the solution, (2) adding any equation to any other equation does not change the solution, (3) switching any two equations does not change the solution. These three operations are equivalent to the elementary row operations described above. The main idea of the Gaussian elimination is to use the elementary row operations to simplify the augmented matrix to its reduced row echelon form. Then, the augmented matrix is further simplified such that its first part is an identity matrix. Given this simplified form, the solution is the last column of the augmented matrix. Let us look at an example:

$$\left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 3 & 5 & 6 & 7 \\ 2 & 4 & 3 & 8 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 2 & 4 & 3 & 8 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 0 & -2 & 7 & -2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & -2 & 7 & -2 \end{array}\right]$$

$$\sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & 0 & 9 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & -15 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right].$$

The last matrix is in reduced row echelon form, and represents the system $x = -15$, $y = 8$, $z = 2$. A comparison with the example in the previous section on the algebraic elimination of variables shows that these two methods are in fact the same; the difference lies in how the computations are written down.

Computational cost of Gaussian elimination

The pseudo code for Gaussian elimination to solve **Ax** = **b**, where **A** is square n-by-n matrix (**A** in assumed to be square for simplicity of the pseudo code below):

```
    // Given matrix A of size n-by-n with elements A[i,j] and column vector b of size n
 1. for k = 1 : n        // can replace n with m in case of m-by-n matrix A
 2.     // Find the k-th pivot:
 3.     i_max  = argmax (i = k ... m, abs(A[i, k])) // argmax finds index of maximal element
 4.     if A[i_max, k] == 0
 5.         error "Matrix is singluar!"     // not full rank, inverse does not exist
 6.     swap rows(k, i_max)
 7.     // Create reduced row-echelon matrix:
 8.     for i = k+1 : n
 9.         // Do for all remaining elements in the current row:
10.         b[i] = b[i] - b[k] * (A[i, k] / A[k, k])
11.         for j = k : n
12.             A[i, j]  := A[i, j] - A[k, j] * (A[i, k] / A[k, k])
13. // Back-substitution:
14. for k = n : -1 : 2    // can replace n with m in case of m-by-n matrix A
15.    // Update b values in rows before row k:
16.    for i = 1 : k-1
17.       b[i] = b[i] - b[k] * A[i, k]
18. // Solution:
19. return b
```

Observe that in lines 2-6 the algorithm first exchanges rows to move the entry with the largest absolute value to the pivot position. Such *partial pivoting* improves the numerical stability of the algorithm; some other variants are used. After lines 7-12, the augmented matrix will be in the reduced row-echelon form. Lines 13-17 perform back-substitution and put the augmented matrix in the simplest form from which the solution can be returned by reading its last column.

Let us study the cost of the shown code: Line 1 is called n times, so it is O(n). Line 3 is called n times. Each call requires finding a maximum element in column k, whose time is O(n). Thus, total cost of line 3 is O(n^2). Cost of lines 4 and 5 is O(1). Since they are called n times, the total cost in O(n^2). Line 6 does row swapping, which takes O(n) time. Total time of line 6 is O(n^2). Cost of Line 8 is also O(n^2). Line 10 is called O(n^2) times and their cost per call is O(1), so their total cost is $O(n^2)$. Lines 11 and 12 are called $O(n^3)$ times and their cost per call is O(1), so their total cost is $O(n^3)$. Lines 14-17 are a double for loop, whose cost is $O(n^2)$. Summing up total costs of all lines we get that the whole code has $O(n^3)$ time.

With modern computers, Gaussian elimination is not always the fastest algorithm to compute the row echelon form of matrix. There are computer libraries, like BLAS, that exploit the specificities of the computer hardware and of the structure of the matrix to automatically choose the best algorithm. This algorithm can be used on a computer for systems with thousands of equations and unknowns. However, the cost becomes prohibitive for systems with millions of equations. These large systems are generally solved using iterative methods. Specific methods exist for systems whose coefficients follow a regular pattern. For example, for sparse systems of linear equations (most elements of **A** are zeros) there are special algorithms that can find solutions in $O(n^2)$ or even O(n) time if **A** is very sparse.

One possible problem is numerical instability, caused by the possibility of dividing by very small numbers. If, for example, the leading coefficient of one of the rows is very close to zero, then to row reduce the matrix one would need to divide by that number so the leading coefficient is 1. This means any error that existed for the number which was close to zero would be amplified. Gaussian elimination is numerically stable for diagonally dominant or positive-definite matrices. For general matrices, Gaussian elimination is usually considered to be stable, when using partial pivoting, even though there are examples of stable matrices for which it is unstable.

Matrix solution

We are jumping ahead here, this paragraph will be clear after learning about the inverse matrix. If the equation system is expressed in the matrix form **Ax = b**, the entire solution set can also be expressed in matrix form. If the matrix **A** is square (has *m* rows and *n*=*m* columns) and has full rank (all *m* rows are independent), then the system has a unique solution given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ where $\mathbf{A}^{-1}$ is the inverse of **A**.

4. Rank of a matrix

In linear algebra, the **rank** of a matrix *A* is the size of the largest collection of linearly independent columns of **A** (the **column rank**) or the size of the largest collection of linearly independent rows of **A** (the **row rank**). For every matrix, the column rank is equal to the row rank. The rank is commonly denoted *rank*(**A**).

The Gaussian elimination algorithm can be applied to any *m*-by-*n* matrix **A**. In this way, for example, some matrices can be transformed to a matrix that has a row echelon form like

$$T = \begin{bmatrix} a & * & * & * & * & * & * & * & * \\ 0 & 0 & b & * & * & * & * & * & * \\ 0 & 0 & 0 & c & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & d & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where the *s are arbitrary entries and *a, b, c, d, e* are nonzero entries. This echelon matrix **T** contains a wealth of information about **A**: the rank of **A** is 5 since there are 5 non-zero rows in **T**.

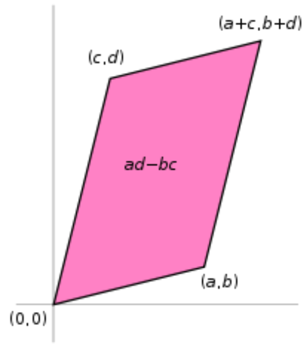
5. Linear transformations

Matrices and matrix multiplication reveal their essential features when related to *linear transformations*, also known as *linear maps*. *A real m-by-n matrix* **A** *gives rise to a linear transformation* $\mathbf{R}^n \to \mathbf{R}^m$ *mapping each vector* $\mathbf{x}$ *in* $\mathbf{R}^n$ *to the (matrix) product* **Ax**, *which is a vector in* $\mathbf{R}^m$. *Conversely, each linear transformation f:* $\mathbf{R}^n \to \mathbf{R}^m$ *arises from a unique m-by-n matrix* **A**: *explicitly, the (i, j)-entry of* **A** *is the* i^{th} *coordinate of* $f(\mathbf{e}_j)$, *where* $\mathbf{e}_j = (0,\ldots,0,1,0,\ldots,0)$ *is the* unit vector *with 1 in the* j^{th} *position and 0 elsewhere.* The matrix **A** is said to represent the linear map *f*, and **A** is called the *transformation matrix* of *f*.

For example, the 2×2 matrix

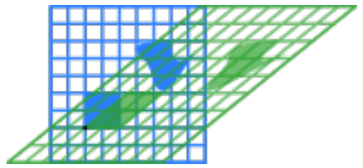
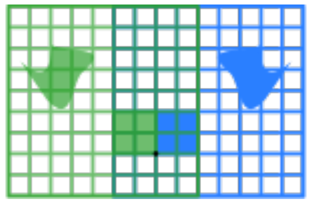
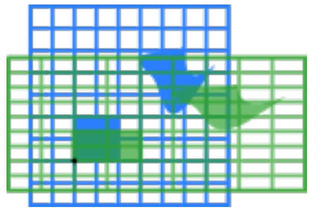
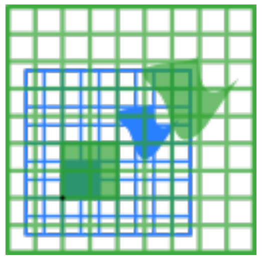
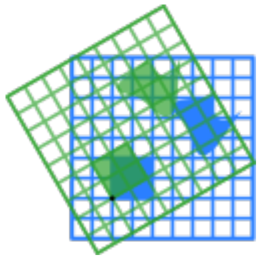
$$\mathbf{A} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

can be viewed as the transform of the unit square into a parallelogram with vertices at (0, 0), (*a*, *b*), (*a* + *c*, *b* + *d*), and (*c*, *d*). The parallelogram pictured at the right is obtained by multiplying **A** with each of the column vectors $\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ in turn. These vectors define the vertices of the unit square.



The vectors represented by a 2-by-2 matrix correspond to the sides of a unit square transformed into a parallelogram.

The following table shows a number of 2-by-2 matrices with the associated linear maps of $\mathbf{R}^2$. The blue original is mapped to the green grid and shapes. The origin (0,0) is marked with a black point.

Horizontal shear with m=1.25.	Horizontal flip	Squeeze mapping with r=3/2	Scaling by a factor of 3/2	*Rotation by π/6^R = 30°*
$\begin{bmatrix}1 & 1.25\\0 & 1\end{bmatrix}$	$\begin{bmatrix}-1 & 0\\0 & 1\end{bmatrix}$	$\begin{bmatrix}3/2 & 0\\0 & 2/3\end{bmatrix}$	$\begin{bmatrix}3/2 & 0\\0 & 3/2\end{bmatrix}$	$\begin{bmatrix}\cos(\pi/6^R) & -\sin(\pi/6^R)\\ \sin(\pi/6^R) & \cos(\pi/6^R)\end{bmatrix}$
				

6. Square matrices

A square matrix is a matrix with the same number of rows and columns. An *n*-by-*n* matrix is known as a square matrix of order *n*. Any two square matrices of the same order can be added and multiplied. The entries a_{ii} form the main diagonal of a square matrix. They lie on the imaginary line which runs from the top left corner to the bottom right corner of the matrix.

Main types

Name	Example with *n* = 3
Diagonal matrix	$\begin{bmatrix}a_{11} & 0 & 0\\0 & a_{22} & 0\\0 & 0 & a_{33}\end{bmatrix}$
Lower triangular matrix	$\begin{bmatrix}a_{11} & 0 & 0\\a_{21} & a_{22} & 0\\a_{31} & a_{32} & a_{33}\end{bmatrix}$

Upper triangular matrix $\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$

The identity matrix $\mathbf{I}_n$ of size n is the n-by-n matrix in which all the elements on the main diagonal are equal to 1 and all other elements are equal to 0, e.g.

$$I_1 = \begin{bmatrix} 1 \end{bmatrix},\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \cdots,\ I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

It is called identity matrix because multiplication with it leaves a matrix unchanged:

$\mathbf{AI}_n = \mathbf{I}_m\mathbf{A} = \mathbf{A}$ for any m-by-n matrix **A**.

A square matrix **A** that is equal to its transpose, i.e., $\mathbf{A} = \mathbf{A}^T$, is a symmetric matrix. If instead, **A** was equal to the negative of its transpose, i.e., $\mathbf{A} = -\mathbf{A}^T$, then **A** is a skew-symmetric matrix.

6.1 Invertible matrix and its inverse

A square matrix **A** is called *invertible* or *non-singular* if there exists a matrix **B** such that

$\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n$.

If **B** exists, it is unique and is called the *inverse matrix* of **A**, denoted $\mathbf{A}^{-1}$.

Finding the inverse of a matrix

A variant of Gaussian elimination called Gauss–Jordan elimination can be used for finding the inverse of a matrix, if it exists. If **A** is a n by n square matrix, then one can use row reduction to compute its inverse matrix, if it exists. First, the n by n identity matrix is augmented to the right of **A**, forming a n by $2n$ block matrix [**A** | **I**]. This is because we are trying to find matrix **X** which is the solution of **A*****X**= **I**. Now through application of elementary row operations, find the reduced echelon form of this n by $2n$ matrix. The matrix **A** is invertible if and only if the left block can be reduced to the identity matrix **I**; in this case the right block of the final matrix is $\mathbf{A}^{-1}$. If the algorithm is unable to reduce the left block to **I**, then **A** is not invertible.

For example, consider the following matrix

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

To find the inverse of this matrix, one takes the following matrix augmented by the identity, and row reduces it as a 3 by 6 matrix:

$$[A|I] = \left[\begin{array}{rrr|rrr} 2 & -1 & 0 & 1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 1 & 0 \\ 0 & -1 & 2 & 0 & 0 & 1 \end{array}\right].$$

By performing row operations, one can check that the reduced row echelon form of this augmented matrix is:

$$[I|B] = \left[\begin{array}{rrr|rrr} 1 & 0 & 0 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 1 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} \end{array}\right].$$

The matrix on the left is the identity, which shows A is invertible. The 3 by 3 matrix on the right, B, is the inverse of A. This procedure for finding the inverse works for square matrices of any size.

6.2. Trace

The trace, tr(**A**) of a square matrix **A** is the sum of its diagonal entries, tr(**A**) = sum(A_{ii}). While matrix multiplication is not commutative as mentioned above, the trace of the product of two matrices is independent of the order of the factors:

tr(**AB**) = tr(**BA**).

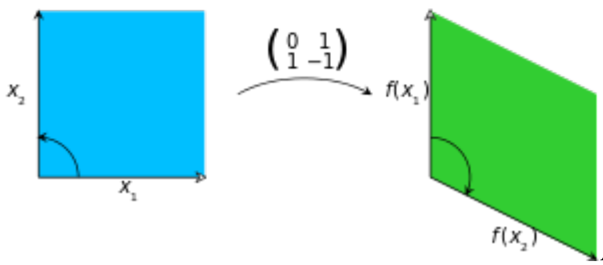
This is immediate from the definition of matrix multiplication:

$$\mathrm{tr}(\mathsf{AB}) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ji} = \mathrm{tr}(\mathsf{BA}).$$

Also, the trace of a matrix is equal to that of its transpose, i.e.,

tr(**A**) = tr(**A**^{T}).

6.3. Determinant



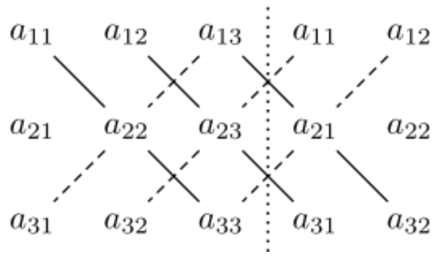
A linear transformation on $\mathbf{R}^2$ given by the indicated matrix. The determinant of this matrix is −1, as the area of the green parallelogram at the right is 1, but the map reverses the orientation, since it turns the counterclockwise orientation of the vectors to a clockwise one.

The *determinant* $det(\mathbf{A})$ or $|\mathbf{A}|$ of a square matrix $\mathbf{A}$ is a number encoding certain properties of the matrix. A matrix is invertible if and only if its determinant is nonzero. Its absolute value equals the area (in $\mathbf{R}^2$) or volume (in $\mathbf{R}^3$) of the image of the unit square (or cube), while its sign corresponds to the orientation of the corresponding linear map: the determinant is positive if and only if the orientation is preserved.

The determinant of 2-by-2 matrices is given by

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc.$$

The determinant of 3-by-3 matrices involves 6 terms (rule of Sarrus):


a_{11} a_{12} a_{13} a_{11} a_{12}
a_{21} a_{22} a_{23} a_{21} a_{22}
a_{31} a_{32} a_{33} a_{31} a_{32}
Sarrus' rule: The determinant of the three columns on the left is the sum of the products along the solid diagonals minus the sum of the products along the dashed diagonals

Laplace's formula and the adjugate matrix

Laplace's formula expresses the determinant of a matrix in terms of its minors. The minor $M_{i,j}$ is defined to be the determinant of the $(n-1) \times (n-1)$-matrix that results from A by removing the ith row and the jth column. The expression $(-1)^{i+j} M_{i,j}$ is known as cofactor. The determinant of A is given by

$$\det(A) = \sum_{j=1}^{n}(-1)^{i+j}a_{i,j}M_{i,j} = \sum_{i=1}^{n}(-1)^{i+j}a_{i,j}M_{i,j}.$$

Calculating det(*A*) by means of that formula is referred to as expanding the determinant along a row or column. For the example 3 × 3 matrix

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

Laplace expansion along the second column (*j* = 2, the sum runs over *i*) yields:

$$\begin{aligned}\det(A) &= (-1)^{1+2}\cdot 2\cdot \begin{vmatrix} -1 & 3 \\ 2 & -1 \end{vmatrix} + (-1)^{2+2}\cdot 1\cdot \begin{vmatrix} -2 & -3 \\ 2 & -1 \end{vmatrix} + (-1)^{3+2}\cdot 0\cdot \begin{vmatrix} -2 & -3 \\ -1 & 3 \end{vmatrix} \\ &= (-2)\cdot((-1)\cdot(-1) - 2\cdot 3) + 1\cdot((-2)\cdot(-1) - 2\cdot(-3)) \\ &= (-2)\cdot(-5) + 8 = 18.\end{aligned}$$

However, Laplace expansion is efficient for small matrices only. The time complexity *f*(*n*) for finding determinant of a matrix of size *n* is *f*(*n*) = *n***f*(*n*-1) = *n*!

Definition of Determinant

There are various ways to define the determinant of a square matrix **A**, i.e. one with the same number of rows and columns. Perhaps the most natural way is expressed in terms of the columns of the matrix. If we write an *n* × *n* matrix in terms of its column vectors

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

where the a_j are vectors of size *n*, then the determinant of **A** is defined so that

$$\begin{aligned}\det\begin{bmatrix} a_1, & \dots, & ba_j + cv, & \dots, a_n \end{bmatrix} &= b\det(A) + c\det\begin{bmatrix} a_1, & \dots, & v, & \dots, a_n \end{bmatrix} \\ \det\begin{bmatrix} a_1, & \dots, & a_j, & a_{j+1}, & \dots, a_n \end{bmatrix} &= -\det\begin{bmatrix} a_1, & \dots, & a_{j+1}, & a_j, & \dots, a_n \end{bmatrix} \\ \det(I) &= 1\end{aligned}$$

where b and c are scalars, v is any vector of size n and I is the identity matrix of size n. Adding a multiple of any row to another row, or a multiple of any column to another column, does not change the determinant. Interchanging two rows or two columns affects the determinant by multiplying it by -1. Using these operations, any matrix can be transformed to a lower (or upper) triangular matrix, and for such matrices the determinant equals the product of the entries on the main diagonal; this provides a method to calculate the determinant of any matrix.

An idea of Gaussian elimination can be used to find determinant of a matrix of size n in $O(n^3)$ time. For example, the determinant of

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

can be computed using the following matrices:

$$B = \begin{bmatrix} -2 & 2 & -3 \\ 0 & 0 & 4.5 \\ 2 & 0 & -1 \end{bmatrix}, \quad C = \begin{bmatrix} -2 & 2 & -3 \\ 0 & 0 & 4.5 \\ 0 & 2 & -4 \end{bmatrix}, \quad D = \begin{bmatrix} -2 & 2 & -3 \\ 0 & 2 & -4 \\ 0 & 0 & 4.5 \end{bmatrix}.$$

Site http://www.math.odu.edu/~bogacki/cgi-bin/lat.cgi can be used to see step by step calculations for matrices that are based on the idea of Gaussian elimination. It shows how reduction to row echelon form works and how it helps in

- Solving a system of linear equations
- Finding an inverse matrix
- Finding a determinant
- Finding a rank

Properties of the determinant

The determinant has many properties. Some basic properties of determinants are:

1. $\det(I_n) = 1$ where In is the $n \times n$ identity matrix.
2. $\det(A^{\mathrm{T}}) = \det(A).$
3. $\det(A^{-1}) = \dfrac{1}{\det(A)} = \det(A)^{-1}.$
4. For square matrices A and B of equal size,

$$\det(AB) = \det(A)\det(B).$$

5. $\det(cA) = c^n \det(A)$ for an $n \times n$ matrix.
6. If A is a triangular matrix, i.e. $ai,j = 0$ whenever $i > j$ or, alternatively, whenever $i < j$, then its determinant equals the product of the diagonal entries:

$$\det(A) = a_{1,1}a_{2,2}\cdots a_{n,n} = \prod_{i=1}^{n} a_{i,i}.$$

A number of additional properties relate to the effects on the determinant of changing particular rows or columns:

7. If in a matrix, any row or column is 0, then the determinant of that particular matrix is 0.
8. Whenever two columns of a matrix are identical, or more generally some column can be expressed as a linear combination of the other columns (i.e. the columns of the matrix form a linearly dependent set), its determinant is 0.

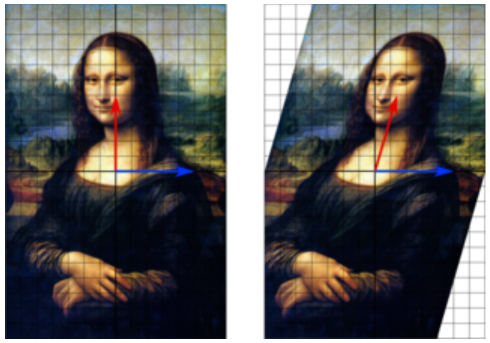
6.4. Eigenvalues and eigenvectors

In linear algebra, an **eigenvector** or **characteristic vector** of a square matrix is a vector that points in a direction which is invariant under the associated linear transformation. In other words, if **v** is a vector which is not zero, then it is an eigenvector of a square matrix A if A**v** is a scalar multiple of **v**. This condition could be written as thge equation

$$A\mathbf{v} = \lambda\mathbf{v}, \qquad (1)$$

where λ is a number (also called a scalar) known as the **eigenvalue** or **characteristic value** associated with the eigenvector **v**.

There is a correspondence between n by n square matrices and linear transformation from an n-dimensional vector space to itself. For this reason, it is equivalent to define eigenvalues and eigenvectors using either the language of matrices or the language of linear transformations.


In this shear mapping the red arrow changes direction but the blue arrow does not. The blue arrow is an eigenvector of this shear mapping because it doesn't change direction, and since in this example its length is unchanged its eigenvalue is 1.

Two dimensional example

Consider the transformation matrix **A**, given by,

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

The figure shows the effect of this transformation on point coordinates in the plane. The eigenvectors **v** of this transformation satisfy the equation,

$$A\mathbf{v} = \lambda\mathbf{v}.$$

Rearrange this equation to obtain

$$(A - \lambda I)\mathbf{v} = 0,$$

which has a solution only when its determinant $|A - \lambda I|$ equals zero.

Set the determinant to zero to obtain the polynomial equation,

$$p(\lambda) = |A - \lambda I| = 3 - 4\lambda + \lambda^2 = 0,$$

known as the characteristic polynomial of the matrix **A**. In this case, it has the roots $\lambda = 1$ and $\lambda = 3$.

For $\lambda = 1$, the equation becomes,

$$(A - I)\mathbf{v} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix},$$

which has the solution,

$$\mathbf{v} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}.$$

For $\lambda = 3$, the equation becomes,

$$(A-3I)\mathbf{w}=\begin{bmatrix}-1 & 1\\ 1 & -1\end{bmatrix}\begin{Bmatrix}w_1\\ w_2\end{Bmatrix}=\begin{Bmatrix}0\\ 0\end{Bmatrix},$$

which has the solution,

$$\mathbf{w}=\begin{Bmatrix}1\\ 1\end{Bmatrix}.$$

Thus, the vectors **v** and **w** are eigenvectors of **A** associated with the eigenvalues $\lambda = 1$ and $\lambda = 3$, respectively.

Diagonal matrices

Matrices with entries only along the main diagonal are called *diagonal matrices*. It is easy to see that the eigenvalues of a diagonal matrix are the diagonal elements themselves. Consider the matrix **A**,

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

The characteristic polynomial of A is given by

$$p(\lambda)=|A-\lambda I|=(1-\lambda)(2-\lambda)(3-\lambda)=0,$$

which has the roots $\lambda = 1$, $\lambda = 2$ and $\lambda = 3$.

Associated with these roots are the eigenvectors,

$$\mathbf{u}=\begin{Bmatrix}1\\ 0\\ 0\end{Bmatrix},\quad \mathbf{v}=\begin{Bmatrix}0\\ 1\\ 0\end{Bmatrix},\quad \text{and}\quad \mathbf{w}=\begin{Bmatrix}0\\ 0\\ 1\end{Bmatrix},$$

respectively.

Triangular matrices

Consider the lower triangular matrix A,

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

The characteristic polynomial of A is given by

$$p(\lambda) = |A - \lambda I| = (1-\lambda)(2-\lambda)(3-\lambda) = 0,$$

which has the roots $\lambda = 1$, $\lambda = 2$ and $\lambda = 3$.

Associated with these roots are the eigenvectors,

$$\mathbf{u} = \begin{Bmatrix} 1 \\ -1 \\ 1/2 \end{Bmatrix}, \quad \mathbf{v} = \begin{Bmatrix} 0 \\ 1 \\ -3 \end{Bmatrix}, \quad \text{and} \quad \mathbf{w} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix},$$

respectively.

Eigenvalues: Real domain

Since the eigenvalues are roots of the characteristic polynomial, an $n \times n$ matrix has at most n distinct eigenvalues. If the matrix has real entries, the coefficients of the characteristic polynomial are all real; but it may have fewer than n real roots, or no real roots at all.

For example, consider the cyclic permutation matrix

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

This matrix shifts the coordinates of the vector up by one position, and moves the first coordinate to the bottom. Its characteristic polynomial is $1 - \lambda^3$ which has only one real root $\lambda_1 = 1$. Any vector with three equal non-zero coordinates is an eigenvector for this eigenvalue. For example,

$$A\begin{bmatrix}5\\5\\5\end{bmatrix}=\begin{bmatrix}5\\5\\5\end{bmatrix}=1\cdot\begin{bmatrix}5\\5\\5\end{bmatrix}$$

Eigenvalues: Complex domain

The fundamental theorem of algebra implies that the characteristic polynomial of an $n \times n$ matrix **A**, being a polynomial of degree n, has exactly n complex roots. More precisely, it can be factored into the product of $n$linear terms,

$$\det(A-\lambda I)=(\lambda_1-\lambda)(\lambda_2-\lambda)\cdots(\lambda_n-\lambda)$$

where each λ_i is a complex number. The numbers λ_1, λ_2, ... λ_n, (which may not be all distinct) are roots of the polynomial, and are precisely the eigenvalues of **A**.

Even if the entries of **A** are all real numbers, the eigenvalues may still have non-zero imaginary parts (and the coordinates of the corresponding eigenvectors will therefore also have non-zero imaginary parts). Also, the eigenvalues may be irrational numbers even if all the entries of **A** are rational numbers, or all are integers.

The non-real roots of a real polynomial with real coefficients can be grouped into pairs of complex conjugate values, namely with the two members of each pair having the same real part and imaginary parts that differ only in sign. If the degree is odd at least one of the roots will be real. Therefore, any real matrix with odd order will have at least one real eigenvalue; whereas a real matrix with even order may have no real eigenvalues.

In the example of the 3×3 cyclic permutation matrix **A**, above, the characteristic polynomial $1-\lambda^3$ has two additional non-real roots, namely

$$\lambda_2=-1/2+\mathbf{i}\sqrt{3}/2,\ \lambda_3=\lambda_2^*=-1/2-\mathbf{i}\sqrt{3}/2,$$

where $\mathbf{i}=\sqrt{-1}$ is the imaginary unit.

Algebraic multiplicity

Let λ_i be an eigenvalue of an $n \times n$ matrix A. The *algebraic multiplicity* $\mu_A(\lambda_i)$ of λ_i is its multiplicity as a root of the characteristic polynomial.

Properties of eigenvalues and eigenvectors

Let **A** be an arbitrary *n*-by-*n* matrix of complex numbers with eigenvalues λ_1, λ_2, ... λ_n. (Here it is understood that an eigenvalue with algebraic multiplicity μ occurs μ times in this list.) Then

- The trace of A, defined as the sum of its diagonal elements, is also the sum of all eigenvalues:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii} = \sum_{i=1}^{n} \lambda_i = \lambda_1 + \lambda_2 + \cdots + \lambda_n.$$

- The determinant of A is the product of all eigenvalues:

$$\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n.$$

- The eigenvalues of the kth power of A, i.e. the eigenvalues of A^k, for any positive integer k, are $\lambda_1^k, \lambda_2^k, \ldots, \lambda_n^k$
- The matrix A is invertible if and only if all the eigenvalues λ_i are nonzero.
- If A is invertible, then the eigenvalues of A^{-1} are $1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_n$. Clearly, the geometric multiplicities coincide. Moreover, since the characteristic polynomial of the inverse is the reciprocal polynomial for that of the original, they share the same algebraic multiplicity.
- If A is a symmetric real matrix then every eigenvalue is real

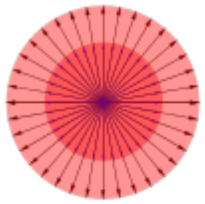
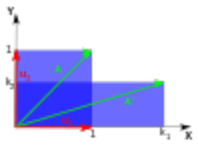
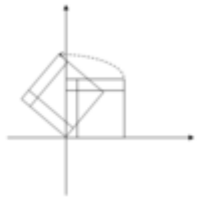
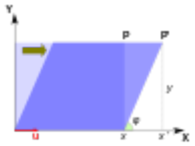
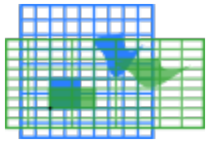
Diagonalization and eigendecomposition

If A has a set of n linearly independent eigenvectors. Let Q be a square matrix whose columns are those eigenvectors, in any order. Then we will have $AQ = Q\Lambda$, where Λ is the diagonal matrix such that Λ_{ii} is the eigenvalue associated to column i of Q. Since the columns of Q are linearly independent, the matrix Q is invertible. Premultiplying both sides by Q^{-1} we get $Q^{-1}AQ = \Lambda$. By definition, therefore, the matrix A is diagonalizable.

If A is diagonalizable, the space of all n-coordinate vectors can be decomposed into the direct sum of the eigenspaces of A. This decomposition is called the eigendecomposition of A, and it is preserved under change of coordinates.

Eigenvalues of geometric transformations

The following table presents some example transformations in the plane along with their 2×2 matrices, eigenvalues, and eigenvectors.

	scaling	unequal scaling	rotation	horizontal shear	hyperbolic rotation
illustration					
matrix	$\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$	$\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$	$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ $c = \cos\theta$ $s = \sin\theta$	$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} c & s \\ s & c \end{bmatrix}$ $c = \cosh\varphi$ $s = \sinh\varphi$
characteristic polynomial	$(\lambda - k)^2$	$(\lambda - k_1)(\lambda - k_2)$	$\lambda^2 - 2c\lambda + 1$	$(\lambda - 1)^2$	$\lambda^2 - 2c\lambda + 1$
eigenvalues λ_i	$\lambda_1 = \lambda_2 = k$	$\lambda_1 = k_1$ $\lambda_2 = k_2$	$\lambda_1 = e^{\mathbf{i}\theta} = c + s\mathbf{i}$ $\lambda_2 = e^{-\mathbf{i}\theta} = c - s\mathbf{i}$	$\lambda_1 = \lambda_2 = 1$	$\lambda_1 = e^{\varphi}$ $\lambda_2 = e^{-\varphi}$,
algebraic multipl. $\mu_i = \mu(\lambda_i)$	$\mu_1 = 2$	$\mu_1 = 1$ $\mu_2 = 1$	$\mu_1 = 1$ $\mu_2 = 1$	$\mu_1 = 2$	$\mu_1 = 1$ $\mu_2 = 1$
geometric multipl. $\gamma_i = \gamma(\lambda_i)$	$\gamma_1 = 2$	$\gamma_1 = 1$ $\gamma_2 = 1$	$\gamma_1 = 1$ $\gamma_2 = 1$	$\gamma_1 = 1$	$\gamma_1 = 1$ $\gamma_2 = 1$
eigenvectors	All non-zero vectors	$u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$u_1 = \begin{bmatrix} 1 \\ -\mathbf{i} \end{bmatrix}$	$u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$u_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad u_2 = \begin{bmatrix} 1 \\ +\mathbf{i} \end{bmatrix} \qquad u_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Note that the characteristic equation for a rotation is a quadratic equation with discriminant $D = -4(\sin\theta)^2$, which is a negative number whenever θ is not an integer multiple of 180°. Therefore, except for these special cases, the two eigenvalues are complex numbers, $\cos\theta \pm \mathbf{i}\sin\theta$; and all eigenvectors have non-real entries. Indeed, except for those special cases, a rotation changes the direction of every nonzero vector in the plane.

Calculation of eigenvalues and eigenvectors

The eigenvalues of a matrix **A** can be determined by finding the roots of the characteristic polynomial. Explicit algebraic formulas for the roots of a polynomial exist only if the degree *n* is 4 or less. Therefore, for matrices of order 5 or more, the eigenvalues and eigenvectors cannot be obtained by an explicit algebraic formula, and must therefore be computed by approximate numerical methods.

In theory, the coefficients of the characteristic polynomial can be computed exactly, since they are sums of products of matrix elements; and there are algorithms that can find all the roots of a polynomial of arbitrary degree to any required accuracy. However, this approach is not viable in practice because the coefficients would be contaminated by unavoidable round-off errors, and the roots of a polynomial can be an extremely sensitive function of the coefficients. Efficient, accurate methods to compute eigenvalues and eigenvectors of arbitrary matrices were not known until the advent of the QR algorithm in 1961. [24]

Once the (exact) value of an eigenvalue is known, the corresponding eigenvectors can be found by finding non-zero solutions of the eigenvalue equation, that becomes a system of linear equations with known coefficients.

6.5. Few other types of square matrices

Definite matrix

A symmetric $n \times n$-matrix is called *positive-definite* (respectively negative-definite; indefinite), if for all nonzero vectors $\mathbf{x} \in \mathbf{R}^n$ the associated quadratic form given by

$$Q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$$

takes only positive values (respectively only negative values; both some negative and some positive values). If the quadratic form takes only non-negative (respectively only non-positive) values, the symmetric matrix is called positive-semidefinite (respectively negative-semidefinite); hence the matrix is indefinite precisely when it is neither positive-semidefinite nor negative-semidefinite.

- If A is also positive-definite, positive-semidefinite, negative-definite, or negative-semidefinite every eigenvalue is positive, non-negative, negative, or non-positive respectively.

Orthogonal matrix

An *orthogonal matrix* is a square matrix with real entries whose columns and rows are orthogonal unit vectors (i.e., orthonormal vectors). Equivalently, a matrix A is orthogonal if its transpose is equal to its inverse:

$$A^{\mathrm{T}} = A^{-1},$$

which entails

$$A^{\mathrm{T}}A = AA^{\mathrm{T}} = I,$$

where I is the identity matrix.

An orthogonal matrix A is necessarily invertible (with inverse $A^{-1} = A^{\mathrm{T}}$). The determinant of any orthogonal matrix is either +1 or −1. A *special orthogonal matrix* is an orthogonal matrix with determinant +1. As a linear transformation, every orthogonal matrix with determinant +1 is a pure rotation, while every orthogonal matrix with determinant -1 is either a pure reflection, or a composition of reflection and rotation.

- Every eigenvalue of a unitary matrix has absolute value $|\lambda| = 1$.

7. Matrix Decompositions

In the mathematical discipline of linear algebra, a **matrix decomposition** or **matrix factorization** is a factorization of a matrix into a product of matrices. There are many different matrix decompositions; each finds use among a particular class of problems.

LU decomposition

- Applicable to: square matrix A
- Decomposition: $A = LU$, where L is lower triangular and U is upper triangular

Cholesky decomposition

- Applicable to: square, symmetric, positive definite matrix A
- Decomposition: $A = U^T U$, where U is upper triangular with positive diagonal entries

Eigendecomposition

- Also called *spectral decomposition*
- Applicable to: square matrix A with distinct eigenvectors (not necessarily distinct eigenvalues).
- Decomposition: $A = VDV^{-1}$, where D is a diagonal matrix formed from the eigenvalues of A, and the columns of V are the corresponding eigenvectors of A.
- Existence: An n-by-n matrix A always has n eigenvalues, which can be ordered (in more than one way) to form an n-by-n diagonal matrix D and a corresponding matrix of nonzero columns V that satisfies the eigenvalue equation $AV = VD$. If the n eigenvectors (not necessarily eigenvalues) are distinct (that is, none is equal to any of the others), then V is invertible, implying the decomposition $A = VDV^{-1}$.

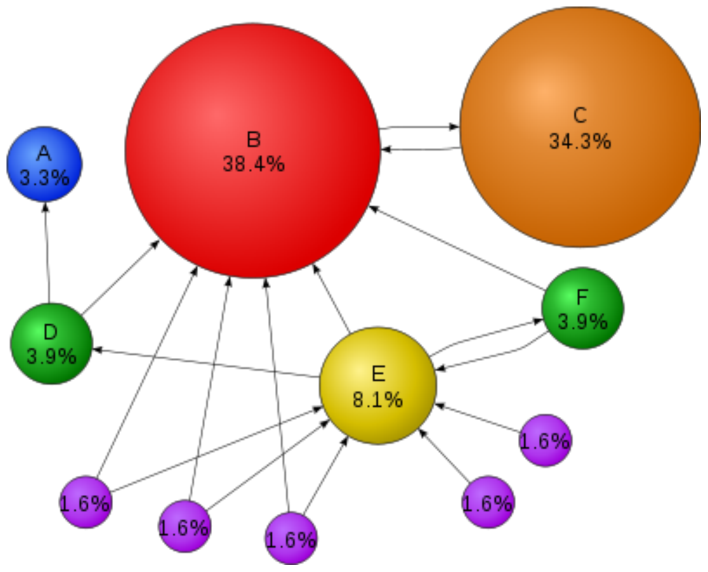
Singular value decomposition

- Applicable to: m-by-n matrix A.
- Decomposition: $A = UDV^H$, where D is a nonnegative diagonal matrix, and U and V are unitary matrices, and V^H denotes the conjugate transpose of V (or simply the transpose, if V contains real numbers only).
- Comment: The diagonal elements of D are called the singular values of A.

8. Applications of Matrices: PageRank

PageRank is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page,[1] one of the founders of Google. PageRank is a way of measuring the importance of website pages. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.[2]

Cartoon illustrating the basic principle of PageRank. The size of each face is proportional to the total size of the other faces which are pointing to it.



Mathematical **PageRanks** for a simple network, expressed as percentages. (Google uses a logarithmic scale.) Page C has a higher PageRank than Page E, even though there are fewer links to C; the one link to C comes from an important page and hence is of high value. If web surfers who start on a random page have an 85% likelihood of choosing a random link from the page they are currently visiting, and a 15% likelihood of jumping to a page chosen at random from the entire web, they will reach Page E 8.1% of the time. (The 15% likelihood of jumping to an arbitrary page corresponds to a damping factor of 85%.) Without damping, all web surfers would eventually end up on Pages A, B, or C, and all other pages would have PageRank zero. In the presence of damping, Page A effectively links to all pages in the web, even though it has no outgoing links of its own.

Main Idea

The main idea of PageRank is that the importance of each page depends on the importance of pages that link to it. Let us denote by $L(v)$ the number of outbound links from page **v**, B_u the set of pages linking to page **u**, and $PR(v)$ the PageRank of page **v**. The PageRank value for any page **u** is expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}, \qquad (*)$$

meaning that the PageRank value for page **u** is dependent on the PageRank values for each page **v** linking to it (i.e., contained in the set B_u), divided by the number $L(v)$ of links from page **v**. Dividing by L(v) allows every page **v** to have the same importance in calculations of PageRanks (without it, the pages with many outbound links would dominate those that have only a few of them). Interestingly, this formulation of the PageRank allows interpreting $PR(u)$ as the probability that an imaginary surfer who is randomly clicking on links will end up at page **u**.

The formula above can be expressed in a matrix form. Let us assume there are N pages, and define **R** as the vector of all N PageRanks

$$R = \begin{bmatrix} PR(u_1) \\ PR(u_2) \\ \dots \\ PR(u_N) \end{bmatrix}.$$

Now, we can rewrite the formula (*) as

$$R = \mathbf{M}R, \qquad (**)$$

where **M** is an *n*-by-*n* adjacency matrix whose elements M_{ij} are defined as

$$M_{ij} = \begin{cases} \frac{1}{L(u_j)}, & \text{if } j \text{ links to } i \\ 0, & \text{otherwise} \end{cases}$$

Observe that matrix **M** is typically very sparse (every page links to only a small fraction of all pages) and that each column of **M** sums up to 1, which means it is a stochastic matrix.

The objective of PageRank is to find *n*-by-1 vector *R* given *n*-by-*n* matrix **M**. After observing that (**) can be rewritten as $\mathbf{M}R = 1R$, we can see that *R* is the eigenvector corresponding to eigenvalue equal to 1 of square matrix **M**.

Damping factor

Solving (**) creates numerical problems and the resulting ranking *R* is not satisfactory. That is why the actual PageRank algorithm modifies (**) as

$$R = d\mathbf{M}R + \frac{1-d}{N}\mathbf{1}, \qquad (***)$$

where **1** is an n-by-1 column of ones, and *d* is a constant between 0 and 1, called the dumping factor, and is typically set to 0.85. What the second term in (***) does is that it gives each page a small weight, regardless of its importance.

In the probabilistic interpretation of (***), we can interpret *PR*(*u*) as the probability that an imaginary surfer who is randomly clicking on links will end up at page *u*. Unlike (**), in (***) the imaginary surfer either continues following the links with probability *d* (= 0.85) or restarts the search by selecting a random page with probability 1 – *d* (= 0.15).

There is an additional technical detail that makes PageRank work well. In practice, there are pages that do not link to any other page. If a random surfer reaches that page, he would get stuck, and this would result in inflating the PageRank of such page. To resolve this, if a page does not link to any other page, PageRank assumes by default that it links to every other page. Intuitively, it means that if a random surfer lands on such a page, he moves to another randomly selected page.

It is interesting to note that various strategies to manipulate PageRank have been employed in concerted efforts to improve search results rankings and monetize advertising links. These strategies have severely impacted the reliability of the PageRank concept, which purports to determine which documents are actually highly valued by the Web community. Since December 2007, when it started *actively* penalizing sites selling paid text links, Google has combatted link farms and other schemes designed to artificially inflate PageRank. How Google identifies link farms and other PageRank manipulation tools is among Google's trade secrets.

Computation

PageRank can be computed either iteratively or algebraically. The iterative method can be viewed as the power iteration method.

Iterative

At $t = 0$ PageRanks of all pages are assumed to be identical, $R = \mathbf{1}/N$. At each time step t, the PageRanks are updated as

$$R^{(t)} = d\mathbf{M}R^{(t-1)} + \frac{1-d}{N}\mathbf{1},$$

The computation ends when for some small ϵ

$$\left| R^{(t)} - R^{(t-1)} \right| < \varepsilon$$

i.e., when convergence is assumed. $|R|$ is a scalar called the norm of vector R and is calculated as $R^{\mathrm{T}}R$.

Algebraic

Formula $R = d\mathbf{M}R + \frac{1-d}{N}\mathbf{1}$ can be rewritten as

$$R = (\mathbf{I} - d\mathbf{M})^{-1}\frac{1-d}{N}\mathbf{1}$$

where $\mathbf{I}$ is the identity matrix. The solution exists and is unique for $0 < d < 1$. This can be seen by noting that $\mathbf{M}$ is by construction a stochastic matrix and hence has an eigenvalue equal to one as a consequence of the Perron–Frobenius theorem.

Power Method

If the matrix $\mathcal{M}$ is a transition probability, i.e., column-stochastic with no columns consisting of just zeros and R is a probability distribution (i.e., $|R| = 1$, $\mathbf{E}R=\mathbf{1}$, where $\mathbf{E}$ is matrix of all ones), Eq. (***) is equivalent to

$$\mathbf{R} = \left(d\mathcal{M} + \frac{1-d}{N}\mathbf{E}\right)\mathbf{R} =: \widehat{\mathcal{M}}\mathbf{R}$$

Hence PageRank *R* is the principal eigenvector of $\widehat{\mathcal{M}}$. A fast and easy way to compute this is using the power method: starting with an arbitrary vector *x*(0), the operator $\widehat{\mathcal{M}}$ is applied in succession, i.e.,

$$x(t+1) = \widehat{\mathcal{M}}x(t),$$

until

$$|x(t+1) - x(t)| < \epsilon.$$

Computational cost and sparse matrices

The adjacency matrix **M** in PageRank applications can be very large. For example, in 2013 Google reported that it indexes 30 trillion ($n = 30*10^{12}$) webpages. Thus, the resulting adjacency matrix **M** has close to 10^{27} elements. Storing this matrix as a two-dimensional array and assuming only one byte per matrix element would require 10^{27} Bytes (this is thousand YottaBytes). Let us assume 1TB (10^{12} B) of hard disk memory costs $1 (it is closer to $50 currently). The cost of hard disc space to hold **M** would be $\$10^{15}$ or 1,000 trillion dollars, which is an order of magnitude larger than the cumulative GDP of the World.

Luckily, adjacency matrix **M** of the Web is very sparse and an average web page links to only a few other web pages. In this case, it is much more efficient to store for each page only the list of pages it links to. Thus, the whole **M** can be stored as an array of such lists. If we assume that the average web page points to 10 other pages and that ID of each page can be stored with 8 Bytes, then the memory needed to store **M** would be $8*10*30*10^{12} = 2.4*10^{15}$ Bytes (or, 2,400 TeraBytes). Assuming $1 price per TeraByte, **M** could be stored with an investment of a few thousand dollars.

In addition to the storage of **M**, PageRank requires matrix operations. The algebraic solution that requires finding an inverse of (**I** – *d***M**) would require $O(n^3)$ time, which in our example would require about 10^{40} calculations and would clearly not be feasible even if all computers of the world are running for billions of years. However, the iterative method is feasible because it consists of repeatedly calculating product **M***R*. Let us consider time to calculate inner product of one row of **M** and *R*. In the average, one row of **M** has only 10 nonzero elements and its inner product with *R* can be calculated in O(1) time. Thus, the product **M***R* would require O(*n*) time which is quite feasible even on a modest computer cluster where it could be done in a matter of seconds by exploiting parallelization.

Solving sparse linear equations

Looking beyond PageRank, there are many applications where there is a need to perform operations on very large matrices. The computational limit of what can be done (even when supercomputers are used) when calculating inverses, determinants, eigenvalues, or matrix decompositions are dense matrices of size up to a million-by-million. However, if matrices are sparse, as the PageRank example illustrates, there is a hope of working with substantially larger matrices. There are numerous methods developed for calculations with sparse matrices. Although it is course on its own, let us illustrate one of the very popular ideas for solving a system of sparse linear equations $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is sparse.

An important special type of sparse matrices is band matrix, defined as follows. The *lower bandwidth* of a matrix $\mathbf{A}$ is the smallest number p such that the entry $a_{i,j}$ vanishes whenever $i > j + p$. Similarly, the *upper bandwidth* is the smallest number p such that $a_{i,j} = 0$ whenever $i < j - p$. For example, a tridiagonal matrix (its main diagonal and one diagonal below and above it contain nonzeros) has lower bandwidth 1 and upper bandwidth 1. As another example, the following sparse matrix has lower and upper bandwidth both equal to 3. Notice that zeros are represented with dots for clarity.

$$\begin{pmatrix} X & X & X & \cdot & \cdot & \cdot & \cdot \\ X & X & \cdot & X & X & \cdot & \cdot \\ X & \cdot & X & \cdot & X & \cdot & \cdot \\ \cdot & X & \cdot & X & \cdot & X & \cdot \\ \cdot & X & X & \cdot & X & X & X \\ \cdot & \cdot & \cdot & X & X & X & \cdot \\ \cdot & \cdot & \cdot & \cdot & X & \cdot & X \end{pmatrix}$$

The extreme example of a band matrix is a diagonal matrix, whose bandwidth equals 1. A very efficient structure for an extreme case of band matrices, the diagonal matrix, is to store just the entries in the main diagonal as a one-dimensional array, so a diagonal $n \times n$ matrix requires only n entries.

Matrices with reasonably small upper and lower bandwidth are known as band matrices and often lend themselves to simpler algorithms than general sparse matrices; or one can sometimes apply dense matrix algorithms and gain efficiency simply by looping over a reduced number of indices.

By rearranging the rows and columns of a matrix $\mathbf{A}$ it may be possible to obtain a matrix $\mathbf{A}'$ with a lower bandwidth. A number of algorithms are designed for bandwidth minimization. If matrix $\mathbf{A}$ is rearranged to band matrix $\mathbf{A}'$, the original problem $\mathbf{Ax} = \mathbf{b}$ can be restated as $\mathbf{A}'\mathbf{x} = \mathbf{b}'$, where $\mathbf{b}'$ is a rearranged vector $\mathbf{b}$ in the same way as $\mathbf{A}$.

If $\mathbf{A}$ is a band matrix with bandwidth p, lines 8-12 of the Gaussian elimination algorithm can be rewritten as

```
8.      for i = k+1 : k+p
9.          // Do for all remaining elements in the current row:
10.         b[i] = b[i] - b[k] * (A[i, k] / A[k, k])
11.         for j = k : k+p
12.             A[i, j]  := A[i, j] - A[k, j] * (A[i, k] / A[k, k])
```

The cost of this piece of code is $O(p^2)$. As a result, the cost of the Gaussian elimination is $O(p^2 n)$. For dense matrices $p = n$, and the cost is $O(n^3)$, for sparse matrices with small fixed p, the cost can be expressed as $O(n)$, which is linear scaling with the problem size!