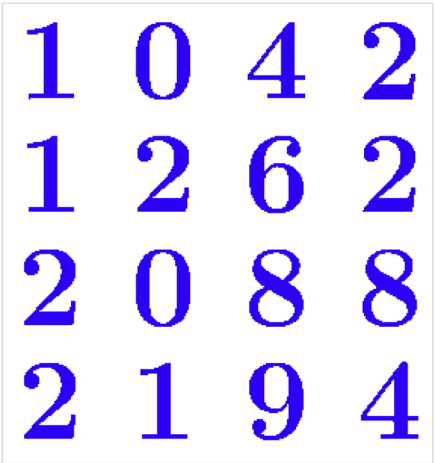
# Gaussian elimination

In mathematics, **Gaussian elimination**, also known as **row reduction**, is an algorithm for solving systems of linear equations. It consists of a sequence of row-wise operations performed on the corresponding matrix of coefficients. This method can also be used to compute the rank of a matrix, the determinant of a square matrix, and the inverse of an invertible matrix. The method is named after Carl Friedrich Gauss (1777–1855). To perform row reduction on a matrix, one uses a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible. There are three types of elementary row operations:



Animation of Gaussian elimination. Red row eliminates the following rows, green rows change their order.

- Swapping two rows,
- Multiplying a row by a nonzero number,
- Adding a multiple of one row to another row.

Using these operations, a matrix can always be transformed into an upper triangular matrix, and in fact one that is in row echelon form. Once all of the leading coefficients (the leftmost nonzero entry in each row) are 1, and every column containing a leading coefficient has zeros elsewhere, the matrix is said to be in reduced row echelon form. This final form is unique; in other words, it is independent of the sequence of row operations used. For example, in the following sequence of row operations (where two elementary operations on different rows are done at the first and third steps), the third and fourth matrices are the ones in row echelon form, and the final matrix is the unique reduced row echelon form.

$$\begin{bmatrix} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 5 & 35 \end{bmatrix} \to \begin{bmatrix} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 2 & 2 & 8 \end{bmatrix} \to \begin{bmatrix} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 0 & 0 & 0 \end{bmatrix} \to \begin{bmatrix} 1 & 0 & -2 & -3 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Using row operations to convert a matrix into reduced row echelon form is sometimes called **Gauss–Jordan elimination**. In this case, the term *Gaussian elimination* refers to the process until it has reached its upper triangular, or (unreduced) row echelon form. For computational reasons, when solving systems of linear equations, it is sometimes preferable to stop row operations before the matrix is completely reduced.

## Definitions and example of algorithm

The process of row reduction makes use of elementary row operations, and can be divided into two parts. The first part (sometimes called forward elimination) reduces a given system to row echelon form, from which one can tell whether there are no solutions, a unique solution, or infinitely many solutions. The second part (sometimes called back substitution) continues to use row operations until the solution is found; in other words, it puts the matrix into reduced row echelon form.

Another point of view, which turns out to be very useful to analyze the algorithm, is that row reduction produces a matrix decomposition of the original matrix. The elementary row operations may be viewed as the multiplication on the left of the original matrix by elementary matrices.

Alternatively, a sequence of elementary operations that reduces a single row may be viewed as multiplication by a Frobenius matrix. Then the first part of the algorithm computes an LU decomposition, while the second part writes the original matrix as the product of a uniquely determined invertible matrix and a uniquely determined reduced row echelon matrix.

## Row operations

There are three types of elementary row operations which may be performed on the rows of a matrix:

1. Swap the positions of two rows.
2. Multiply a row by a non-zero scalar.
3. Add to one row a scalar multiple of another.

If the matrix is associated to a system of linear equations, then these operations do not change the solution set. Therefore, if one's goal is to solve a system of linear equations, then using these row operations could make the problem easier.

## Echelon form

For each row in a matrix, if the row does not consist of only zeros, then the leftmost nonzero entry is called the *leading coefficient* (or *pivot*) of that row. So if two leading coefficients are in the same column, then a row operation of type 3 could be used to make one of those coefficients zero. Then by using the row swapping operation, one can always order the rows so that for every non-zero row, the leading coefficient is to the right of the leading coefficient of the row above. If this is the case, then matrix is said to be in row echelon form. So the lower left part of the matrix contains only zeros, and all of the zero rows are below the non-zero rows. The word "echelon" is used here because one can roughly think of the rows being ranked by their size, with the largest being at the top and the smallest being at the bottom.

For example, the following matrix is in row echelon form, and its leading coefficients are shown in red:

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

It is in echelon form because the zero row is at the bottom, and the leading coefficient of the second row (in the third column), is to the right of the leading coefficient of the first row (in the second column).

A matrix is said to be in reduced row echelon form if furthermore all of the leading coefficients are equal to 1 (which can be achieved by using the elementary row operation of type 2), and in every column containing a leading coefficient, all of the other entries in that column are zero (which can be achieved by using elementary row operations of type 3).

## Example of the algorithm

Suppose the goal is to find and describe the set of solutions to the following system of linear equations:

$$\begin{aligned} 2x + y - z &= 8 && (L_1) \\ -3x - y + 2z &= -11 && (L_2) \\ -2x + y + 2z &= -3 && (L_3) \end{aligned}$$

The table below is the row reduction process applied simultaneously to the system of equations and its associated augmented matrix. In practice, one does not usually deal with the systems in terms of equations, but instead makes use of the augmented matrix, which is more suitable for computer manipulations. The row reduction procedure may be summarized as follows: eliminate $x$ from all equations below $L_1$, and then eliminate $y$ from all equations below $L_2$. This will put the system into triangular form. Then, using back-substitution, each unknown can be solved for.

| System of equations | Row operations | Augmented matrix |
|---|---|---|
| $\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$ | | $\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$ |
| $\begin{aligned} 2x + y - z &= 8 \\ \tfrac{1}{2}y + \tfrac{1}{2}z &= 1 \\ 2y + z &= 5 \end{aligned}$ | $\begin{aligned} L_2 + \tfrac{3}{2}L_1 &\to L_2 \\ L_3 + L_1 &\to L_3 \end{aligned}$ | $\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{array}\right]$ |
| $\begin{aligned} 2x + y - z &= 8 \\ \tfrac{1}{2}y + \tfrac{1}{2}z &= 1 \\ -z &= 1 \end{aligned}$ | $L_3 + -4L_2 \to L_3$ | $\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$ |
| The matrix is now in echelon form (also called triangular form) | | |
| $\begin{aligned} 2x + y &= 7 \\ \tfrac{1}{2}y &= \tfrac{3}{2} \\ -z &= 1 \end{aligned}$ | $\begin{aligned} L_1 - L_3 &\to L_1 \\ L_2 + \tfrac{1}{2}L_3 &\to L_2 \end{aligned}$ | $\left[\begin{array}{ccc\|c} 2 & 1 & 0 & 7 \\ 0 & \frac{1}{2} & 0 & \frac{3}{2} \\ 0 & 0 & -1 & 1 \end{array}\right]$ |
| $\begin{aligned} 2x + y &= 7 \\ y &= 3 \\ z &= -1 \end{aligned}$ | $\begin{aligned} 2L_2 &\to L_2 \\ -L_3 &\to L_3 \end{aligned}$ | $\left[\begin{array}{ccc\|c} 2 & 1 & 0 & 7 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$ |
| $\begin{aligned} x &= 2 \\ y &= 3 \\ z &= -1 \end{aligned}$ | $\begin{aligned} L_1 - L_2 &\to L_1 \\ \tfrac{1}{2}L_1 &\to L_1 \end{aligned}$ | $\left[\begin{array}{ccc\|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$ |

The second column describes which row operations have just been performed. So for the first step, the $x$ is eliminated from $L_2$ by adding $\frac{3}{2}L_1$ to $L_2$. Next, $x$ is eliminated from $L_3$ by adding $L_1$ to $L_3$. These row operations are labelled in the table as

$$\begin{aligned} L_2 + \tfrac{3}{2}L_1 &\to L_2, \\ L_3 + L_1 &\to L_3. \end{aligned}$$

Once $y$ is also eliminated from the third row, the result is a system of linear equations in triangular form, and so the first part of the algorithm is complete. From a computational point of view, it is faster to solve the variables in reverse order, a process known as back-substitution. One sees the solution is $z = -1$, $y = 3$, and $x = 2$. So there is a unique solution to the original system of equations.

Instead of stopping once the matrix is in echelon form, one could continue until the matrix is in *reduced* row echelon form, as it is done in the table. The process of row reducing until the matrix is reduced is sometimes referred to as Gauss–Jordan elimination, to distinguish it from stopping after reaching echelon form.

# History

The method of Gaussian elimination appears – albeit without proof – in the Chinese mathematical text Chapter Eight: *Rectangular Arrays* of *The Nine Chapters on the Mathematical Art*. Its use is illustrated in eighteen problems, with two to five equations. The first reference to the book by this title is dated to 179 AD, but parts of it were written as early as approximately 150 BC.[1][2][3] It was commented on by Liu Hui in the 3rd century.

The method in Europe stems from the notes of Isaac Newton.[4][5] In 1670, he wrote that all the algebra books known to him lacked a lesson for solving simultaneous equations, which Newton then supplied. Cambridge University eventually published the notes as *Arithmetica Universalis* in 1707 long after Newton had left academic life. The notes were widely imitated, which made (what is now called) Gaussian elimination a standard lesson in algebra textbooks by the end of the 18th century. Carl Friedrich Gauss in 1810 devised a notation for symmetric elimination that was adopted in the 19th century by professional hand computers to solve the normal equations of least-squares problems.[6] The algorithm that is taught in high school was named for Gauss only in the 1950s as a result of confusion over the history of the subject.[7]

Some authors use the term *Gaussian elimination* to refer only to the procedure until the matrix is in echelon form, and use the term Gauss–Jordan elimination to refer to the procedure which ends in reduced echelon form. The name is used because it is a variation of Gaussian elimination as described by Wilhelm Jordan in 1888. However, the method also appears in an article by Clasen published in the same year. Jordan and Clasen probably discovered Gauss–Jordan elimination independently.[8]

# Applications

Historically, the first application of the row reduction method is for solving systems of linear equations. Below are some other important applications of the algorithm.

## Computing determinants

To explain how Gaussian elimination allows the computation of the determinant of a square matrix, we have to recall how the elementary row operations change the determinant:

- Swapping two rows multiplies the determinant by −1
- Multiplying a row by a nonzero scalar multiplies the determinant by the same scalar
- Adding to one row a scalar multiple of another does not change the determinant.

If Gaussian elimination applied to a square matrix $A$ produces a row echelon matrix $B$, let $d$ be the product of the scalars by which the determinant has been multiplied, using the above rules. Then the determinant of $A$ is the quotient by $d$ of the product of the elements of the diagonal of $B$:

$$\det(A) = \frac{\prod \operatorname{diag}(B)}{d}.$$

Computationally, for an $n \times n$ matrix, this method needs only $O(n^3)$ arithmetic operations, while using Leibniz formula for determinants requires $(n\,n!)$ operations (number of summands in the formula times the number of multiplications in each summand), and recursive Laplace expansion requires $O(n\ 2^n)$ operations if the sub-determinants are memorized for being computed only once (number of operations in a linear combination times the number of sub-determinants to compute, which are determined by their columns). Even on the fastest computers, these two methods are impractical or almost impracticable for $n$ above 20.

## 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 an $n \times n$ square matrix, then one can use row reduction to compute its inverse matrix, if it exists. First, the $n \times n$ identity matrix is augmented to the right of $A$, forming an $n \times 2n$ block matrix $[A \mid I]$. Now through application of elementary row operations, find the reduced echelon form of this $n \times 2n$ 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 $A^{-1}$. 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 $3 \times 6$ matrix:

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

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

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

One can think of each row operation as the left product by an elementary matrix. Denoting by $B$ the product of these elementary matrices, we showed, on the left, that $BA = I$, and therefore, $B = A^{-1}$. On the right, we kept a record of $BI = B$, which we know is the inverse desired. This procedure for finding the inverse works for square matrices of any size.

## Computing ranks and bases

The Gaussian elimination algorithm can be applied to any $m \times n$ matrix $A$. In this way, for example, some 6 × 9 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 stars 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 nonzero rows in $T$; the vector space spanned by the columns of $A$ has a basis consisting of its columns 1, 3, 4, 7 and 9 (the columns with $a$, $b$, $c$, $d$, $e$ in $T$), and the stars show how the other columns of $A$ can be written as linear combinations of the basis columns.

All of this applies also to the reduced row echelon form, which is a particular row echelon format.

# Computational efficiency

The number of arithmetic operations required to perform row reduction is one way of measuring the algorithm's computational efficiency. For example, to solve a system of $n$ equations for $n$ unknowns by performing row operations on the matrix until it is in echelon form, and then solving for each unknown in reverse order, requires $n(n + 1)/2$ divisions, $(2n^3 + 3n^2 - 5n)/6$ multiplications, and $(2n^3 + 3n^2 - 5n)/6$ subtractions,[9] for a total of approximately $2n^3/3$ operations. Thus it has a *arithmetic complexity* (time complexity, where each arithmetic operation take a unit of time, independently of the size of the inputs) of $O(n^3)$.

This complexity is a good measure of the time needed for the whole computation when the time for each arithmetic operation is approximately constant. This is the case when the coefficients are represented by floating-point numbers or when they belong to a finite field. If the coefficients are integers or rational numbers exactly represented, the intermediate entries can grow exponentially large, so the bit complexity is exponential.[10] However, Bareiss' algorithm is a variant of Gaussian elimination that avoids this exponential growth of the intermediate entries; with the same arithmetic complexity of $O(n^3)$, it has a bit complexity of $O(n^5)$, and has therefore a strongly-polynomial time complexity.

Gaussian elimination and its variants can be used on computers 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 (see system of linear equations).

## Bareiss algorithm

The first strongly-polynomial time algorithm for Gaussian elimination was published by Jack Edmonds in 1967.[11]:37 Independently, and almost simultaneously, Erwin Bareiss discovered another algorithm, based on the following remark, which applies to a division-free variant of Gaussian elimination.

In standard Gaussian elimination, one subtracts from each row $R_i$ below the pivot row $R_k$ a multiple of $R_k$ by $r_{i,k}/r_{k,k}$, where $r_{i,k}$ and $r_{k,k}$ are the entries in the pivot column of $R_i$ and $R_k$, respectively.

Bareiss variant consists, instead, of replacing $R_i$ with $\frac{r_{k,k}R_i - r_{i,k}R_k}{r_{k-1,k-1}}$. This produces a row echelon form that has the same zero entries as with the standard Gaussian elimination.

Bareiss' main remark is that each matrix entry generated by this variant is the determinant of a submatrix of the original matrix.

In particular, if one starts with integer entries, the divisions occurring in the algorithm are exact divisions resulting in integers. So, all intermediate entries and final entries are integers. Moreover, Hadamard inequality provides an upper bound on the absolute values of the intermediate and final entries, and thus a bit complexity of $\tilde{O}(n^5)$, using soft O notation.

Moreover, as an upper bound on the size of final entries is known, a complexity $\tilde{O}(n^4)$ can be obtained with modular computation followed either by Chinese remaindering or Hensel lifting.

As a corollary, the following problems can be solved in strongly polynomial time with the same bit complexity:[11]:40

- Testing whether *m* given rational vectors are linearly independent
- Computing the determinant of a rational matrix
- Computing a solution of a rational equation system $Ax = b$
- Computing the inverse matrix of a nonsingular rational matrix
- Computing the rank of a rational matrix

### Numeric instability

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. This means that any error which existed for the number that 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.[12]

## Generalizations

Gaussian elimination can be performed over any field, not just the real numbers.

Buchberger's algorithm is a generalization of Gaussian elimination to systems of polynomial equations. This generalization depends heavily on the notion of a monomial order. The choice of an ordering on the variables is already implicit in Gaussian elimination, manifesting as the choice to work from left to right when selecting pivot positions.

Computing the rank of a tensor of order greater than 2 is NP-hard.[13] Therefore, if P ≠ NP, there cannot be a polynomial time analog of Gaussian elimination for higher-order tensors (matrices are array representations of order-2 tensors).

# Pseudocode

As explained above, Gaussian elimination transforms a given $m \times n$ matrix $A$ into a matrix in row-echelon form.

In the following pseudocode, `A[i, j]` denotes the entry of the matrix $A$ in row $i$ and column $j$ with the indices starting from 1. The transformation is performed *in place*, meaning that the original matrix is lost for being eventually replaced by its row-echelon form.

```
h := 1 /* Initialization of the pivot row */
k := 1 /* Initialization of the pivot column */

while h ≤ m and k ≤ n
    /* Find the k-th pivot: */
    i_max := argmax (i = h ... m, abs(A[i, k]))
    if A[i_max, k] = 0
        /* No pivot in this column, pass to next column */
        k := k + 1
    else
        swap rows(h, i_max)
        /* Do for all rows below pivot: */
        for i = h + 1 ... m:
            f := A[i, k] / A[h, k]
            /* Fill with zeros the lower part of pivot column: */
            A[i, k] := 0
            /* Do for all remaining elements in current row: */
            for j = k + 1 ... n:
                A[i, j] := A[i, j] - A[h, j] * f
        /* Increase pivot row and column */
        h := h + 1
        k := k + 1
```

This algorithm differs slightly from the one discussed earlier, by choosing a pivot with largest absolute value. Such a *partial pivoting* may be required if, at the pivot place, the entry of the matrix is zero. In any case, choosing the largest possible absolute value of the pivot improves the numerical stability of the algorithm, when floating point is used for representing numbers.[14]

Upon completion of this procedure the matrix will be in row echelon form and the corresponding system may be solved by back substitution.

# See also

- Fangcheng (mathematics)
- Gram–Schmidt process - another process for bringing a matrix into some canonical form.

# References


1. "DOCUMENTA MATHEMATICA, Vol. Extra Volume: Optimization Stories (2012), 9-14" (https://www.emis.de/journals/DMJDMV/vol-ismp/10_yuan-ya-xiang.html). *www.emis.de*. Retrieved 2022-12-02.
2. Calinger 1999, pp. 234–236

3. Timothy Gowers; June Barrow-Green; Imre Leader (8 September 2008). *The Princeton Companion to Mathematics* (https://archive.org/details/princetoncompanio00gowe). Princeton University Press. p. 607. ISBN 978-0-691-11880-2.
4. Grcar 2011a, pp. 169–172
5. Grcar 2011b, pp. 783–785
6. Lauritzen, p. 3
7. Grcar 2011b, p. 789
8. Althoen, Steven C.; McLaughlin, Renate (1987), "Gauss–Jordan reduction: a brief history", *The American Mathematical Monthly*, **94** (2), Mathematical Association of America: 130–142, doi:10.2307/2322413 (https://doi.org/10.2307%2F2322413), ISSN 0002-9890 (https://www.worldcat.org/issn/0002-9890), JSTOR 2322413 (https://www.jstor.org/stable/2322413)
9. Farebrother 1988, p. 12
10. Fang, Xin Gui; Havas, George (1997). "On the worst-case complexity of integer Gaussian elimination" (https://scholar.archive.org/work/2htta67odrfilhxegzjqfratyy). *Proceedings of the 1997 international symposium on Symbolic and algebraic computation*. ISSAC '97. Kihei, Maui, Hawaii, United States: ACM. pp. 28–31. doi:10.1145/258726.258740 (https://doi.org/10.1145%2F258726.258740). ISBN 0-89791-875-4.
11. Grötschel, Martin; Lovász, László; Schrijver, Alexander (1993), *Geometric algorithms and combinatorial optimization* (https://books.google.com/books?id=hWvmCAAAQBAJ&pg=PA281), Algorithms and Combinatorics, vol. 2 (2nd ed.), Springer-Verlag, Berlin, doi:10.1007/978-3-642-78240-4 (https://doi.org/10.1007%2F978-3-642-78240-4), ISBN 978-3-642-78242-8, MR 1261419 (https://mathscinet.ams.org/mathscinet-getitem?mr=1261419)
12. Golub & Van Loan 1996, §3.4.6
13. Hillar, Christopher; Lim, Lek-Heng (2009-11-07). "Most tensor problems are NP-hard". arXiv:0911.1393 (https://arxiv.org/abs/0911.1393) [cs.CC (https://arxiv.org/archive/cs.CC)].
14. Kurgalin, Sergei; Borzunov, Sergei (2021). "Algebra and Geometry with Python" (https://doi.org/10.1007/978-3-030-61541-3). *SpringerLink*. doi:10.1007/978-3-030-61541-3 (https://doi.org/10.1007%2F978-3-030-61541-3).

## Works cited


- Atkinson, Kendall A. (1989), *An Introduction to Numerical Analysis* (2nd ed.), New York: John Wiley & Sons, ISBN 978-0471624899.
- Bolch, Gunter; Greiner, Stefan; de Meer, Hermann; Trivedi, Kishor S. (2006), *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications* (2nd ed.), Wiley-Interscience, ISBN 978-0-471-79156-0.
- Calinger, Ronald (1999), *A Contextual History of Mathematics*, Prentice Hall, ISBN 978-0-02-318285-3.
- Farebrother, R.W. (1988), *Linear Least Squares Computations*, STATISTICS: Textbooks and Monographs, Marcel Dekker, ISBN 978-0-8247-7661-9.
- Lauritzen, Niels, *Undergraduate Convexity: From Fourier and Motzkin to Kuhn and Tucker*.
- Golub, Gene H.; Van Loan, Charles F. (1996), *Matrix Computations* (3rd ed.), Johns Hopkins, ISBN 978-0-8018-5414-9.
- Grcar, Joseph F. (2011a), "How ordinary elimination became Gaussian elimination", *Historia Mathematica*, **38** (2): 163–218, arXiv:0907.2397 (https://arxiv.org/abs/0907.2397), doi:10.1016/j.hm.2010.06.003 (https://doi.org/10.1016%2Fj.hm.2010.06.003), S2CID 14259511 (https://api.semanticscholar.org/CorpusID:14259511)
- Grcar, Joseph F. (2011b), "Mathematicians of Gaussian elimination" (https://www.ams.org/notices/201106/rtx110600782p.pdf) (PDF), *Notices of the American Mathematical Society*, **58** (6): 782–792
- Higham, Nicholas (2002), *Accuracy and Stability of Numerical Algorithms* (2nd ed.), SIAM, ISBN 978-0-89871-521-7.
- Katz, Victor J. (2004), *A History of Mathematics, Brief Version*, Addison-Wesley, ISBN 978-0-321-16193-2.

- Kaw, Autar; Kalu, Egwu (2010). "Numerical Methods with Applications: Chapter 04.06 Gaussian Elimination" (http://mathforcollege.com/nm/mws/gen/04sle/mws_gen_sle_txt_gaussian.pdf) (PDF) (1st ed.). University of South Florida. Archived (https://web.archive.org/web/20120907224427/http://mathforcollege.com/nm/mws/gen/04sle/mws_gen_sle_txt_gaussian.pdf) (PDF) from the original on 2012-09-07.
- Lipson, Marc; Lipschutz, Seymour (2001), *Schaum's outline of theory and problems of linear algebra*, New York: McGraw-Hill, pp. 69–80, ISBN 978-0-07-136200-9
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 2.2" (https://web.archive.org/web/20120319193237/http://apps.nrbook.com/empanel/index.html?pg=46), *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8, archived from the original (http://apps.nrbook.com/empanel/index.html?pg=46) on 2012-03-19, retrieved 2011-08-08


# External links

- Interactive didactic tool (http://pnp.mathematik.uni-stuttgart.de/igt/eiserm/lehre/Gael/)