Gaussian elimination

From Wikipedia, the free encyclopedia

In linear algebra, **Gaussian elimination** is an efficient algorithm for solving systems of linear equations, finding the rank of a matrix, and calculating the inverse of an invertible square matrix. Gaussian elimination is named after German mathematician and scientist Carl Friedrich Gauss.

Elementary row operations are used to reduce a matrix to row echelon form. An extension of this algorithm, Gauss–Jordan elimination, reduces the matrix further to reduced row echelon form. Gaussian elimination alone is sufficient for many applications.

Contents

- 1 History
- 2 Algorithm overview
- 3 Example
- 4 Other applications
  - 4.1 Finding the inverse of a matrix
  - 4.2 General algorithm to compute ranks and bases
- 5 Analysis
  - 5.1 Higher order tensors
- 6 Pseudocode
- 7 See also
- 8 Notes
- 9 References
- 10 External links

History

The method of Gaussian elimination appears in Chapter Eight, *Rectangular Arrays*, of the important Chinese mathematical text *Jiuzhang suanshu* or *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 CE, but parts of it were written as early as approximately 150 BCE.[1] It was commented on by Liu Hui in the 3rd century.

However, the method was invented in Europe independently by Carl Friedrich Gauss when developing the method of least squares in his 1809 publication *Theory of Motion of Heavenly Bodies*.[2]

Algorithm overview

The process of Gaussian elimination has two parts. The first part (Forward Elimination) reduces a given system to either triangular or echelon form, or results in a degenerate equation with no solution, indicating the system has no solution. This is accomplished through the use of elementary row operations. The second step uses back substitution to find the solution of the system above.

Stated equivalently for matrices, the first part reduces a matrix to row echelon form using elementary row operations while the second reduces it to reduced row echelon form, or row canonical form.

Another point of view, which turns out to be very useful to analyze the algorithm, is that Gaussian elimination computes a matrix decomposition. The three elementary row operations used in the Gaussian elimination (multiplying rows, switching rows, and adding multiples of rows to other rows) amount to multiplying the original matrix with invertible matrices from the left. 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.

Example

Suppose the goal is to find and describe the solution(s), if any, of 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 algorithm is 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.

In our example, we eliminate *x* from L_2 by adding $\frac{3}{2}L_1$ to L_2, and then we eliminate *x* from L_3 by adding L_1 to L_3. Formally:

$$\begin{aligned} L_2 + \frac{3}{2}L_1 &\rightarrow L_2 \\ L_3 + L_1 &\rightarrow L_3 \end{aligned}$$

The result is:

$$\begin{aligned} 2x + y - z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ 2y + z &= 5 \end{aligned}$$

Now we eliminate *y* from L_3 by adding $-4L_2$ to L_3:

$$L_3 + -4L_2 \rightarrow L_3$$

The result is:

$$\begin{aligned} 2x + y - z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ -z &= 1 \end{aligned}$$

This result is a system of linear equations in triangular form, and so the first part of the algorithm is complete.

The second part, back-substitution, consists of solving for the unknowns in reverse order. Thus, we can easily see that

$$z = -1 \quad (L_3)$$

Then, *z* can be substituted into L_2, which can then be solved easily to obtain

$$y = 3 \quad (L_2)$$

Next, *z* and *y* can be substituted into L_1, which can be solved to obtain

$$x = 2 \quad (L_1)$$

Thus, the system is solved.

Some systems cannot be reduced to triangular form, yet still have at least one valid solution: for example, if *y* had not occurred in L_2 and L_3 after our first step above, the algorithm would have been unable to reduce the system to triangular form. However, it would still have reduced the system to echelon form. In this case, the system does not have a unique solution, as it contains at least one free variable. The solution set can then be expressed parametrically (that is, in terms of the free variables, so that if values for the free variables are chosen, a solution will be generated).

In practice, one does not usually deal with the actual systems in terms of equations but instead makes use of the augmented matrix (which is also suitable for computer manipulations). This, then, is the Gaussian Elimination algorithm applied to the augmented matrix of the system above, beginning with:

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

which, at the end of the first part of the algorithm looks like this:

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$$

That is, it is in row echelon form.

At the end of the algorithm, if the Gauss–Jordan elimination is applied we are left with

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$$

That is, it is in reduced row echelon form, or row canonical form.

Other applications

Finding the inverse of a matrix

Suppose A is a $n \times n$ matrix and you need to calculate its inverse. The $n \times n$ identity matrix is augmented to the right of A, forming a $n \times 2n$ matrix (the block matrix $B = [A,I]$). Through application of elementary row operations and the Gaussian elimination algorithm, the left block of B can be reduced to the identity matrix I, which leaves A^{-1} in the right block of B.

If the algorithm is unable to reduce A to triangular form, then A is not invertible.

In practice, inverting a matrix is rarely required. Most of the time, one is really after the solution of a particular system of linear equations.[3]

General algorithm to compute ranks and bases

The Gaussian elimination algorithm can be applied to any $m \times n$ matrix A. If we get "stuck" in a given column, we move to the next column. In this way, for example, some 6×9 matrices can be transformed to a matrix that has a reduced row echelon form like

$$\begin{bmatrix} 1 & * & 0 & 0 & * & * & 0 & * & 0 \\ 0 & 0 & 1 & 0 & * & * & 0 & * & 0 \\ 0 & 0 & 0 & 1 & * & * & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(the *'s are arbitrary 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; the vector space spanned by the columns of A has a basis consisting of the first, third, fourth, seventh and ninth column of A (the columns of the ones in T), and the *'s tell you how the other columns of A can be written as linear combinations of the basis columns.

Analysis

Gaussian elimination to solve a system of n equations for n unknowns requires $n(n+1) / 2$ divisions, $(2n^3 + 3n^2 - 5n)/6$ multiplications, and $(2n^3 + 3n^2 - 5n)/6$ subtractions,[4] for a total of approximately $2n^3 / 3$ operations. So it has a complexity of $\mathcal{O}(n^3)$.

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 (see system of linear equations).

The Gaussian elimination can be performed over any field.

Gaussian elimination is numerically stable for diagonally dominant or positive-definite matrices. For general matrices, Gaussian elimination is usually considered to be stable in practice if you use partial pivoting as described below, even though there are examples for which it is unstable.[5]

Higher order tensors

Gaussian elimination does not generalize in any simple way to higher order tensors (matrices are order 2 tensors); even computing the rank of a tensor of order greater than 2 is a difficult problem.

Pseudocode

As explained above, Gaussian elimination writes a given $m \times n$ matrix A uniquely as a product of an invertible $m \times m$ matrix S and a row-echelon matrix T. Here, S is the product of the matrices corresponding to the row operations performed.

The formal algorithm to compute T from A follows. We write $A[i,j]$ for the entry in row i, column j in matrix A. The transformation is performed "in place", meaning that the original matrix A is lost and successively replaced by T.

```
i := 1
j := 1
while (i ≤ m and j ≤ n) do
  Find pivot in column j, starting in row i:
  maxi := i
  for k := i+1 to m do
    if abs(A[k,j]) > abs(A[maxi,j]) then
      maxi := k
    end if
  end for
  if A[maxi,j] ≠ 0 then
    swap rows i and maxi, but do not change the value of i
    Now A[i,j] will contain the old value of A[maxi,j].
    divide each entry in row i by A[i,j]
    Now A[i,j] will have the value 1.
    for u := i+1 to m do
      subtract A[u,j] * row i from row u
      Now A[u,j] will be 0, since A[u,j] - A[i,j] * A[u,j] = A[u,j] - 1 * A[u,j] = 0.
    end for
    i := i + 1
  end if
  j := j + 1
end while
```

This algorithm differs slightly from the one discussed earlier, because before eliminating a variable, it 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 variants are also in use.

The column currently being transformed is called the pivot column. Proceed from left to right, letting the pivot column be the first column, then the second column, etc. and finally the last column before the vertical line. For each pivot column, do the following two steps before moving on to the next pivot column:

1. Locate the diagonal element in the pivot column. This element is called the pivot. The row containing the pivot is called the pivot row. Divide every element in the pivot row by the pivot to get a new pivot row with a 1 in the pivot position.
2. Get a 0 in each position below the pivot position by subtracting a suitable multiple of the pivot row from each of the rows below it.

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

See also

- Frontal solver
- Gauss-Jordan elimination
- Grassmann's algorithm is a numerically stable variant of Gaussian elimination. The algorithm avoids subtractions so is less sensitive to errors caused by subtraction of two roughly equal numbers.[6]

Notes

1. ^ Calinger, pp 234–236
2. ^ Katz, §18.1.2
3. ^ Atkinson (1989), p. 514
4. ^ Farebrother, R.W. (1988). *Linear Least Squares Computations*. STATISTICS: Textbooks and Monographs. Marcel Dekker Inc.. pp. 12. ISBN 0824776615.
5. ^ Golub and Van Loan, §3.4.6
6. ^ Bolch, Gunter; Greiner, Stefan; de Meer, Hermann; Trivedi, Kishor S. (2006). *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications* (2 ed.). Wiley-Interscience. p. 158. ISBN 0471791563, 9780471791560.

References

- Atkinson, Kendall A. *An Introduction to Numerical Analysis*, 2nd edition, John Wiley & Sons, New York, 1989. ISBN 0-471-50023-2.
- Calinger, Ronald (1999), *A Contextual History of Mathematics*, Prentice Hall, ISBN 978-0-02-318285-3.
- Golub, Gene H., and Van Loan, Charles F. *Matrix computations*, 3rd edition, Johns Hopkins, Baltimore, 1996. ISBN 0-8018-5414-8.
- Katz, Victor J. (2004), *A History of Mathematics, Brief Version*, Addison-Wesley, ISBN 978-0-321-16193-2.
- Lipschutz, Seymour, and Lipson, Mark. *Schaum's Outlines: Linear Algebra*. Tata McGraw-hill edition.Delhi 2001. pp. 69-80.

External links

- A program that performs Gaussian elimination similarly to a human working on paper (http://marekrychlik.com/cgi-bin/gauss.cgi) It produces exact solutions to systems with rational coefficients. It also understands complex rational numbers. Works with floating point numbers, too.
- Gaussian elimination (http://www.math-linux.com/spip.php?article53) www.math-linux.com.
- Gaussian elimination as java applet (http://www25.brinkster.com/denshade/GaussElimination.html) at some local site. Only takes natural coefficients.
- Gaussian elimination (http://numericalmethods.eng.usf.edu/topics/gaussian_elimination.html) at Holistic Numerical Methods Institute (http://numericalmethods.eng.usf.edu)
- LinearEquations.c (http://www.hlevkin.com/default.html#numalg) Gaussian elimination implemented using C language

Retrieved from "http://en.wikipedia.org/wiki/Gaussian_elimination"
Categories: Numerical linear algebra | Articles with example pseudocode

- This page was last modified on 18 May 2009 at 15:34.
- Text is available under the Creative Commons Attribution/Share-Alike License; additional terms may apply. See Terms of Use for details. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a U.S. registered 501(c)(3) tax-deductible nonprofit charity.