International Journal of Engineering, Science and Mathematics
Vol. 9 Issue 12, December 2020,
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

STUDY OF LINEAR EQUATION ON ALGEBRAIC METHOD WITH CONSTANT VARIABLES

Monika*, Dr. Vijesh Kumar**
Research Scholar*, Assistant Professor**
Singhania University Pacheri Bari Jhunjhunu, Rajasthan

ABSTRACT:
The algebraic method refers to various methods of solving a pair of linear equations, including graphing, substitution and elimination.

The graphing method involves graphing the two equations. The intersection of the two lines will be an x,y coordinate, which is the solution.

With the substitution method, rearrange the equations to express the value of variables, x or y, in terms of another variable. Then substitute that expression for the value of that variable in the other equation.

KEYWORDS-algebraic, graphing, coordinate, variable, expression

INTRODUCTION:

Constant is a value that cannot be reassigned. A constant is immutable and cannot be changed. There are some methods in different programming languages to define a constant variable. Most uses the **const** keyword. Using a **const** keyword indicates that the data is read-only. We can use the **const** keyword in programming languages such as C, C++, JavaScript, PHP, etc. It can be applied to declare an object. Java does not directly support the constants. The alternative way to define the constants in java is by using the non-static modifiers **static** and **final**.[1,2]

The keyword **const** creates a read-only reference to the value. It can neither be re-declared nor can't a value be reassigned to it. That means a const variable or its value can't be changed in a program.

Linear equations are equations of the first order. The linear equations are defined for lines in the coordinate system. When the equation has a homogeneous variable of degree 1 (i.e. only one variable), then it is known as a linear equation in one variable. A linear equation can have more than one variable. If the linear equation has two variables, then it is called linear equations in two variables and so on. Some of the examples of linear equations are $2x – 3 = 0$, $2y = 8$, $m + 1 = 0$, $x/2 = 3$, $x + y = 2$, $3x – y + z = 3$. In this article, we are going to discuss the definition of linear equations, standard form for linear equation in one variable, two variables, three variables and their examples with complete explanation.
An equation is a mathematical statement, which has an equal sign (=) between the algebraic expression. Linear equations are the equations of degree 1. It is the equation for the straight line. The solutions of linear equations will generate values, which when substituted for the unknown values, make the equation true. In the case of one variable, there is only one solution. For example, the equation $x + 2 = 0$ has only one solution as x =

-2. But in the case of the two-variable linear equation, the solutions are calculated as the Cartesian coordinates of a point of the Euclidean plane.[3,4]

Below are some examples of linear equations in one variable, two variables and three variables:

Linear Equation in One variable	**Linear Equation in Two variables**	**Linear Equation in Three variables**
3x+5=0 (3/2)x +7 = 0 **98x = 49**	**y+7x=3** **3a+2b = 5** **6x+9y-12=0**	**x + y + z = 0** **a – 3b = c** **3x + 12 y = ½ z**

Forms of Linear Equation

The three forms of linear equations are

- Standard Form
- Slope Intercept Form
- Point Slope Form

Now, let us discuss these three major forms of linear equations in detail.

Standard Form of Linear Equation

Linear equations are a combination of constants and variables. The standard form of a linear equation in one variable is represented as

ax + b = 0, where, $a \neq 0$ and x is the variable.

The standard form of a linear equation in two variables is represented as

ax + by + c = 0, where, $a \neq 0$, $b \neq 0$, x and y are the variables.

The standard form of a linear equation in three variables is represented as

ax + by + cz + d = 0, where $a \neq 0$, $b \neq 0$, $c \neq 0$, x, y, z are the variables.

Slope Intercept Form

The most common form of linear equations is in slope-intercept form, which is represented as;

y = mx + b

Where,

m is the slope of the line,

b is the y-intercept

x and y are the coordinates of the x-axis and y-axis, respectively.

For example, $y = 3x + 7$:

slope, $m = 3$ and intercept = 7

If a straight line is parallel to the x-axis, then the x-coordinate will be equal to zero. Therefore,

$y=b$

If the line is parallel to the y-axis then the y-coordinate will be zero.[5,6]

$mx+b = 0$

$x=-b/m$

Slope: The slope of the line is equal to the ratio of the change in y-coordinates to the change in x-coordinates. It can be evaluated by:

$m = (y_2-y_1)/(x_2-x_1)$

So basically the slope shows the rise of line in the plane along with the distance covered in the x-axis. The slope of the line is also called a gradient.

Point Slope Form

In this form of linear equation, a straight line equation is formed by considering the points in the x-y plane, such that:

$\mathbf{y - y_1 = m(x - x_1)}$

where (x_1, y_1) are the coordinates of the point.

We can also express it as:

$y = mx + y_1 - mx_1$

Summary:

There are different forms to write linear equations. Some of them are:

Linear Equation	**General Form**	**Example**
Slope intercept form	y = mx + b	y + 2x = 3
Point–slope form	$y - y_1 = m(x - x_1)$	y – 3 = 6(x – 2)
General Form	Ax + By + C = 0	2x + 3y – 6 = 0
Intercept form	x/a + y/b = 1	x/2 + y/3 = 1
As a Function	f(x) instead of y f(x) = x + C	f(x) = x + 3
The Identity Function	f(x) = x	f(x) = 3x
Constant Functions	f(x) = C	f(x) = 6

Where m = slope of a line; (a, b) intercept of x-axis and y-axis.

How to Solve Linear Equations?

By now you have got an idea of linear equations and their different forms. Now let us learn how to solve linear equations or line equations in one variable, in two variables and in three variables with examples. Solving these equations with step by step procedures are given here.

Solution of Linear Equations in One Variable

Both sides of the equation are supposed to be balanced for solving a linear equation. The equality sign denotes that the expressions on either side of the 'equal to' sign are equal. Since the equation is balanced, for solving it, certain mathematical operations are performed on both sides of the equation in a manner that does not affect the balance of the equation. Here is the example related to the linear equation in one variable.[2,3]

Example: Solve **(2x – 10)/2 = 3(x – 1)**

Step 1: Clear the fraction

x – 5 = 3(x – 1)

Step 2: Simplify Both sides equations

x – 5 = 3x – 3

x = 3x + 2

Step 3: Isolate x

$x - 3x = 2$

$-2x = 2$

$x = -1$

Solution of Linear Equations in Two Variables

To solve linear equations in 2 variables, there are different methods. Following are some of them:

1. Method of substitution
2. Cross multiplication method
3. Method of elimination

We must choose a set of 2 equations to find the values of 2 variables. Such as $ax + by + c = 0$ and $dx + ey + f = 0$, also called a system of equations with two variables, where x and y are two variables and a, b, c, d, e, f are constants, and a, b, d and e are not zero. Else, the single equation has an infinite number of solutions.

Solution of Linear Equations in Three Variables

To solve linear equations in 3 variables, we need a set of 3 equations as given below to find the values of unknowns. Matrix method is one of the popular methods to solve system of linear equations with 3 variables.

$a_1x + b_1 y + c_1z + d_1 = 0$

$a_2x + b_2 y + c_2 z + d_2 = 0$ and

$a_3x + b_3 y + c_3 z + d_3 = 0$

Solving Linear Equations

Example 1:

Solve $x = 12(x + 2)$

Solution:

$x = 12(x + 2)$

$x = 12x + 24$

Subtract 24 on both sides of equation

$x - 24 = 12x + 24 - 24$

$x - 24 = 12x$

Simplify

$11x = -24$

Isolate x:

x = -24/11.

Example 2:

Solve x – y = 12 and 2x + y = 22

Solution:

Name the equations

x – y = 12 …(1)

2x + y = 22 …(2)

Isolate Equation (1) for x,

x = y + 12

Substitute x =y + 12 in equation (2)

2(y+12) + y = 22

3y + 24 = 22

3y = -2

or y = -2/3

Substitute the value of y in x = y + 12

x = y + 12

x = -2/3 + 12

x = 34/3

Answer: x = 34/3 and y = -2/3

DISCUSSION:

The algebraic method discussed earlier for testing/finding points of intersection applies, of course, to ellipses since they are implicitly defined by quadratic equations. In some applications, more information is needed other than just knowing points of intersection. Specifically, if the ellipses are used as bounding regions, it might be important to know if one ellipse is fully contained in another. This information is not provided by the algebraic method applied to the two quadratic equations defining the ellipses. The more precise queries for ellipses E_0 and E_1 are

- Do E_0 and E_1 intersect?
- Are E_0 and E_1 separated? That is, does there exist a line for which the ellipses are on opposite sides?
- Is E_0 properly contained in E_1, or is E_1 properly contained in E_0?

Let the ellipse E_i be defined by the quadratic equation $Q_i(X) = X^T A_i X + B_i^T X + c_i$ for $i = 0, 1$. It is assumed that the A_i are positive definite. In this case, $Q_i(X) < 0$ defines the inside of the ellipse, and $Q_i(X) > 0$ defines the outside.

The discussion focuses on level curves of the quadratic functions. Section A.9.1 provides a discussion of level sets of functions. All level curves defined by $Q_0(x, y) = \lambda$ are ellipses, except for the minimum (negative) value λ for which the equation defines a single point, the center of every level curve ellipse. The ellipse defined by $Q_1(x, y) = 0$ is a curve that generally intersects many level curves of Q_0.[3,4] The problem is to find the minimum level value λ_0 and maximum level value λ_1 attained by any (x, y) on the ellipse E. If $\lambda_1 < 0$, then E_1 is properly contained in E_0. If $\lambda_0 > 0$, then E_0 and E_1 are separated or E_1 contains E_0. Otherwise, $0 \in [\lambda_0, \lambda_1]$ and the two ellipses intersect.

RESULTS:

When no suitable combinatorial or algebraic method is available for constructing a code with given parameters, we can instead use computer search techniques. The exhaustive search is seldom feasible, but there are a number of *local search* methods that can be employed. In this section we briefly discuss some of them, in particular simulated annealing and taboo search, and how they can be applied to finding good covering codes.

Usually the parameters q, n, K and r are fixed before the search. The objective is to find an $(n, K)R$ code with $R \leq r$. If such a code is found we decrease K and try to find a smaller one.

When using the *iterative improvement* method we start with K – usually randomly chosen – q-ary vectors of length n. Such a configuration is a code, if all the K vectors are different. For each configuration we define its *neighbourhood,* which, e.g., consists of all the configurations that can be obtained by changing at most r coordinates in one vector. At each step we change the configuration to a randomly chosen one in its neighbourhood. If a better configuration is obtained, the change is accepted, otherwise not. The same process is then continued until no further improvements are possible. The quality of a configuration is measured by a *cost function.* A natural choice is to define the cost function as the number of vectors in the whole space that are not r-covered by the K vectors of the current configuration. We can use an array with q^n entries to store the number of times each q-ary vector of length n is covered. Updating this array and calculating the cost of the new configuration is easy after each change. The goal is to try to minimize the cost function value. Indeed, a configuration with cost 0 has covering radius at most r. This method is simple and efficient, but liable to get stuck in local minima. In the *steepest descent* method,

instead of choosing a random neighbour, we go through the whole neighbourhood and choose the configuration with the least cost.
In *simulated annealing* we sometimes accept changes that deteriorate the configuration, thus hoping to escape from local minima. This method mimics physical annealing, slow cooling of material from a high-energy liquid state to a low-energy crystallized state. In simulated annealing, we first choose a high initial temperature T which is slowly lowered, typically by multiplying the previous temperature by a constant λ, where $0.9 \leq \lambda < 1$. At each temperature we generate a number of new configurations, e.g., a constant number. Each new configuration with a lower cost is accepted. A configuration with a higher cost is accepted with a probability
At first when T is high, this probability is large, but decreases when T is lowered. Eventually, we find a configuration with cost 0, i.e., a configuration with covering radius at most r, or anyway stop, if the cost remains the same for a fixed number of consecutive temperatures. This basic variant can be modified in a number of ways.
In *taboo search* we go through the whole neighbourhood and accept the neighbour with the smallest cost – even if this means increasing the cost. However, after accepting a configuration with a higher cost we do not wish to slide back to the local minimum in the next move. To avoid it we keep a *taboo list* of forbidden moves. In our case a taboo list may consist of the L most recently altered vectors in the configuration, and it is forbidden to change a vector which is in the taboo list. Then L denotes the number of moves that must take place before an altered vector can be altered again.
Another natural neighbourhood structure is obtained as follows. We go through the vectors in the whole space one by one, e.g., in the lexicographic order, until we find an uncovered vector $\mathbf{x}$. The neighbourhood consists of all configurations that can be obtained by replacing exactly one vector in the current configuration by a vector that r-covers $\mathbf{x}$.
Local search can also be used in connection with the matrix method of Section 3.5 to search for the matrix $\mathbf{A}$ and the set S. First, the parameters q, n, r, k and $|S|$ are fixed. A suitable cost function is the number of vectors in Q^k that cannot be written in the form $\mathbf{Ay} + \mathbf{s}$ for any $\mathbf{y} \in Q^n$ of weight at most r and $\mathbf{s} \in S$. During the search we may change both $\mathbf{A}$ and S. Alternatively, we can change only S, but go through some large collection of matrices $\mathbf{A}$ or choose a particularly promising matrix $\mathbf{A}$, e.g., a parity check matrix of a good linear code. Linear covering codes can be searched by fixing $S = \{00 \dots 0\}$.[4,5]

CONCLUSION:

Algebraic methods (sometimes called analytic) provide an equalization solution in a finite number of operations, and can always be employed as judicious initializations to iterative equalizers. An algebraic CM solution is obtained in [25], where the CM criterion is formulated as a nonlinear least squares *(LS)* problem. Through an appropriate transformation of the equalizer parameter space, the nonlinear system becomes a linear LS problem subject to certain constraints on the solution structure. Recovering of the correct structure is particularly important when multiple *zero forcing (ZF)* solutions exist; for instance, in all-pole channels with over-parameterized finite impulse response *(FIR)* equalizers, several ZF equalization delays are possible. From a matrix algebra perspective, enforcing this structure can be considered as a matrix diagonalization problem, where the transformation matrix is composed of the equalizer vectors. Once a non-structured solution has been obtained via pseudo-inversion, the minimum-length equalizer can be extracted by a subspace-based approach or other simple procedures for structure restoration.
The blind equalization method of [25] has strong connections with the *analytical* CM algorithm *(ACMA)* of [82] for BSS. ACMA yields, in the noiseless case, exact algebraic

solutions for the spatial filters extracting the sources from observed instantaneous linear mixtures. It is interesting to note that the recovery of separating spatial filters from a basis of the solution space is equivalent to the joint diagonalization of the corresponding matrices. This joint diagonalization can be performed by the generalized *Schur decomposition* [32] of several (more than two) matrices, for which a convergence proof has not yet been found. Either for source separation or channel equalization, ACMA requires special modifications to treat signals with one-dimensional alphabets (e.g. binary) [25,81,82]. Such modifications give rise to the *real ACMA (RACMA)* method [81].
Other solutions aiming at estimating algebraically the best SISO equalizer, or to identify the SISO channel, when the input belongs to a known alphabet, have been proposed in The *discrete alphabet* hypothesis is then crucial, and replaces the assumption of statistical independence between symbols [14], which is no longer necessary. The alphabet-based CP criterion also admits algebraic solutions, which, as reviewed in section 15.6, can be considered as a generalization of the algebraic CM solutions. Algebraic CP solutions are linked to challenging tensor decomposition problems. For a symbol constellation, the minimum-length equalizer can be determined from the joint decomposition of th-order tensors, which, in turn, is linked to the rank-1 linear combination problem in the tensor case. To surmount the lack of effective tools for performing this task, approximate solutions can be proposed in the form of a subspace method exploiting the particular structure of the tensors associated with satisfactory equalization solutions. As opposed to [5], the subspace method proposed here takes into account a complete basis of the solution space. The use of this additional information allows one to increase the robustness of the algorithm with respect to the structure of the minimum-length equalizer. Moreover, the proposed blind algebraic solution deals naturally with binary inputs (BPSK, MSK) without any modifications.[5,6]

REFERENCES:

1. Definition of "Algebraic function" in the Encyclopedia of Math
2. Weisstein, Eric W. "Algebraic Function". MathWorld.
3. Algebraic Function at PlanetMath.
4. Definition of "Algebraic function" Archived 2020-10-26 at the Wayback Machine in David J. Darling's Internet Encyclopedia of Science
5. Ahlfors, Lars *(1979). Complex Analysis. McGraw Hill.*
6. van der Waerden, B.L. *(1931). Modern Algebra, Volume II. Springer.*
7. Tedeev Al. F. Finitnostnositelyaresheniyazadachi v oblastitipaoktanta. Vestnik VGU. Seriyafizika. matematika. 2014. №4. p. 1-14
8. Eidus D., Katin S. The filtration equation in class of functions decreasing at infinity. Proceedings of the American Mathematical Society. 1994. vol. 120. № 3. p. 825-830
9. Dorothee. D. Horoske. Sobolev spaces with muckenhoupt weights, singularities and inequalities. Georgian math. 2008. v. 15. № 2. p. 263-280
10. Katin S., Kersner R. Disapperance of interfaces in finite time. Mechanica. 1983. v. 28. p. 117-120
11. Adams R. Sobolev spaces. New York. Academic Press, 1975. p. 429
12. Tedeev A. F. The interface flow-up phenomenon and local estimates for doubly degenerate parabolic equations. Applicable analysis. 2007. v. 86. № 6. p. 756-782
13. Vazques I. l. The porous medium equation. Clarendon Press. 2007. p. 586