**Introduction**

In data science and machine learning, we often represent data as vectors and matrices. In mathematics and physics, vectors are defined as quantities that capture a magnitude and a direction (e.g., a distance vector). For example, we may represent data on a person’s demographic information (e.g., race, age, gender, etc.) as a vector, yet there is not a pure geometric interpretation of magnitude or direction. Similarly, matrices, in mathematics, are meant to represent linear mappings, which is defined as the mapping between two vector spaces that preserve vector addition and scalar multiplication. Yet, the context with how matrices are used in data science / ML is different from this formal mathematical definition.

When working with data, we often want to manipulate them and / or feed them into machine learning models. This process involves a lot of computation and will often require adding and multiplying many numbers. For example, in building a recommendation system for movies, you might collect data on how long users viewed each movie in the library. Then you can recommend movies that on average have a higher watch time (as that might translate into better engagement). This average is calculated for each movie by adding all the watch time across all the users and dividing by the number of users. Doing this process element wise can be slow, especially when the number of users and movies get high (such as the case for Netflix which has well over 100 million subscribers and has over thousands of titles).

However, computer scientists have developed extremely efficient algorithms for linear algebra. Addition and multiplication with vectors and matrices are much faster than conventional element wise addition / multiplication. For Python, the NumPy library, which is for scientific computing and linear algebra, provides faster speed and efficiency. Re-visiting our recommendation system problem, we can think of each user being associated with a vector of watch times with dimensions n, where n is the number of movies. Then our data will be a matrix collection of these vectors, with n rows and m columns, where n is the number of movies and m is the number of users. To find movies to recommend, we can average along the rows to find the average watch time for each movie across all users, then sort the movies by the highest average watch time. Implementing this problem with vectors and matrices allows for faster computation because of highly optimized algorithms.

**Vectors**

**Geometric and Coordinate Vectors**

You can distinguish *geometric vectors*, which are arrows pointing in space, from *coordinate vectors*, which are list of values stored in arrays. The relationship between the two is that you can take the coordinates of the endpoint of the arrows to get values that depends on the coordinate system.

Mathematically, you can refer to vectors with lowercase bold italic letters, as $boldsymbol{***v***}$ for instance. Let’s have the following vector $boldsymbol{***v***}$:

With Numpy, vectors are coordinate vectors: one-dimensional arrays of numerical values. You can create vectors with the function np.array():

import numpy as np

npv=np.array([1,-1])  
varray([ 1, -1])

The variable v contains a Numpy one-dimensional array, that is, a vector, containing two values. From a geometric point of view, you can consider each of these values as coordinates. Since there are only two values, you can represent the vector in a Cartesian plane.

# Using Vectors with Numpy

You saw how to create a vector using the function array(). Note also that many Numpy functions return arrays. For instance, look at the following chunk of code:

np.random.seed(123)  
random\_vector=np.random.normal(0,1,2)  
random\_vectorarray([-1.0856306 , 0.99734545])

The function np.random.normal() is used to draw random values from a normal distribution. You can see that it returns a Numpy array with the random values

# Indexing

np.random.seed(123)  
b=np.random.normal(0,1,10)  
b=array([-1.0856306 , 0.99734545, 0.2829785 , -1.50629471, -0.57860025,1.65143654,-2.42667924,-0.42891263,1.26593626,0.8667404 ])

You can get only part of the vector using indexing. You can use values or list of values as indexes. For instance:

b[0]

-1.0856306033005612

b[[0, 2]]

array([-1.0856306, 0.2829785])

You can also use a semicolon to get element from an index to another: start:end. For example:

b[2:5]

array([ 0.2829785 , -1.50629471, -0.57860025])

If you omit start or end, it will uses respectively the first element and the last element. For instance, [:] will return the three first elements.

b[:3]array([-1.0856306 , 0.99734545, 0.2829785 ])

You can index from the last value using a negative sign. For instance, -1 corresponds to the last value, -2 to the one before, etc.:

b[-1]

-0.8667404022651017

b[-2]

1.265936258705534

You can also look at the shape of an array with the attribute shape:

b.shape(10,)

You can see that there are 10 components in the vector $b$. Looking at the shape of your vectors tells you how many components it contains.

# Indexing

Like with vectors, you can get subsets of matrices using indexing. Since there are rows and columns, you need to use two indexes. For instance, remembering that Python uses zero-based indexing, to get the elements in the second row and the third column of the preceding matrix $boldsymbol{***C***}$, you do:

C[1, 2]

1.651436537097151

If you want to get the column 0, you need to take all rows (using :) for this column:

C[:, 0]

array([-1.0856306 , -1.50629471, -2.42667924, -0.8667404 , 1.49138963])

If you want the last rows, you can do the same (all columns using :) and use -1:

C[-1, :]

array([ 1.49138963, -0.638902 , -0.44398196])

# Matrices

Say that you have multiple vectors corresponding to different observations from your dataset. You have one vector per observation with a length corresponding to the number of features. Similarly, you can have one vector per features corresponding containing each observations (you’ll see that transposition allows you to go from one view to another).

Matrices are two-dimensional arrays: they have rows and columns. You can denote a matrix with an uppercase bold italic letter, as $boldsymbol{***A***}$. For instance, you can have:

$$ boldsymbol{***A***} = begin{bmatrix} 1 & 2 3 & 4 5 & 6 end{bmatrix} $$

The matrix $boldsymbol{***A***}$ contains three rows and two columns. You can think of it as two column vectors or as three row vectors.

Let’s take an example creating a matrix containing random values:

np.random.seed(123)  
C=np.random.normal(0,1,(5,3))  
C=array([[-1.0856306 , 0.99734545, 0.2829785 ],  
 [-1.50629471, -0.57860025, 1.65143654],  
 [-2.42667924, -0.42891263, 1.26593626],  
 [-0.8667404 , -0.67888615, -0.09470897],  
 [ 1.49138963, -0.638902 , -0.44398196]])

The matrix $boldsymbol{***C***}$ has 5 rows and 3 columns. You can look at its shape using again the shape attribute:

C.shape(5, 3)

**Solving Linear equations**

All types of programming use mathematics at some level. Machine learning involves programming data to learn the function that best describes the data.

The problem (or process) of finding the best parameters of a function using data is called **model training**in ML.

Therefore, in a nutshell, machine learning is programming to optimize for the best possible solution – and we need math to understand how that problem is solved.

The first step towards learning Math for ML is to learn linear algebra.

Linear Algebra is the mathematical foundation that solves the problem of representing data as well as computations in machine learning models.

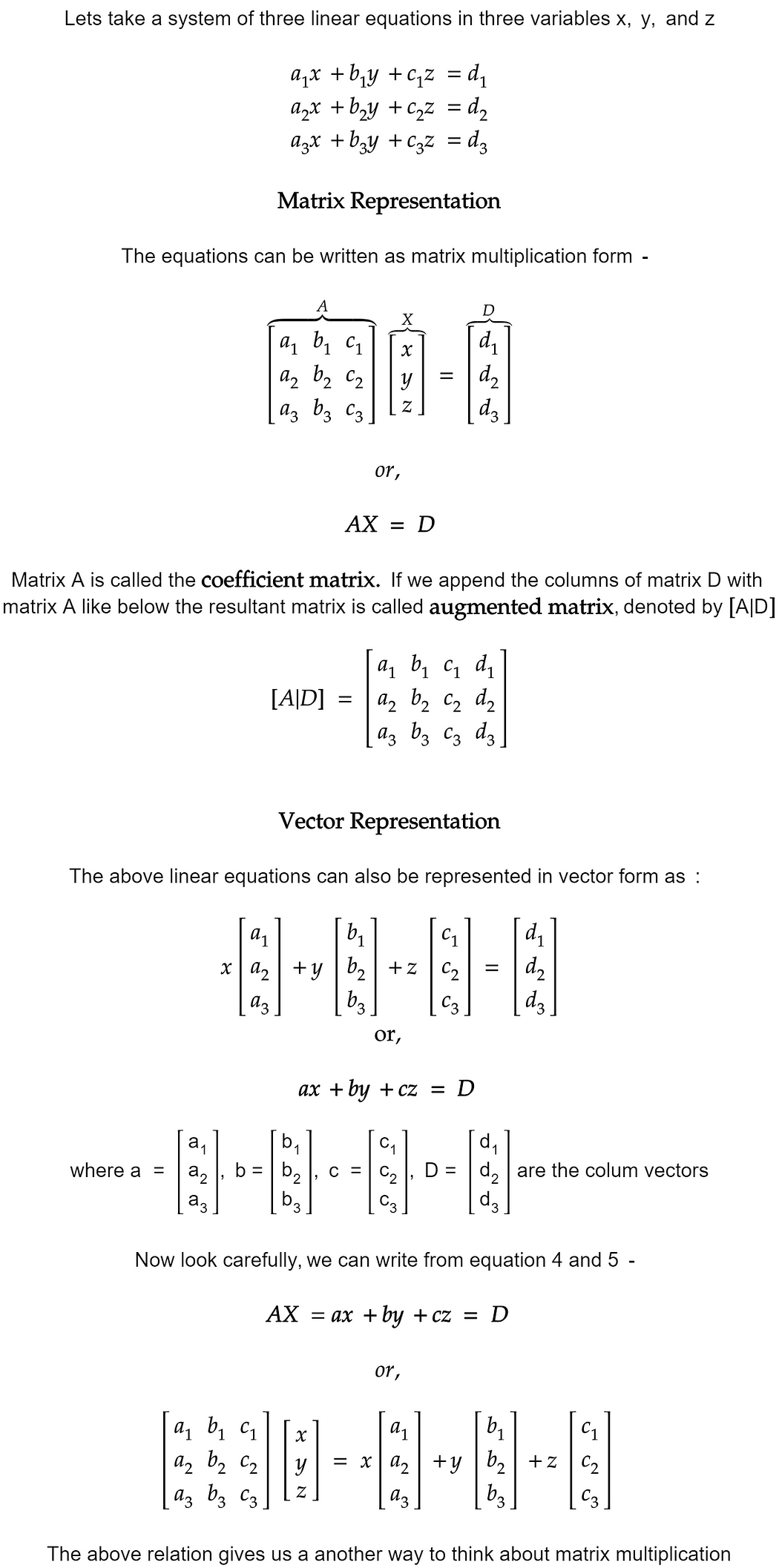
In the ML context, all major phases of developing a model have linear algebra running behind the scenes.

Important areas of application that are enabled by linear algebra are:

* data and learned model representation
* word embeddings
* dimensionality reduction

*A****system of linear equations****(or****linear system****) is a collection of two or more linear equations involving the same set of variables.*

Representation of linear equations in matrix and vector forms:



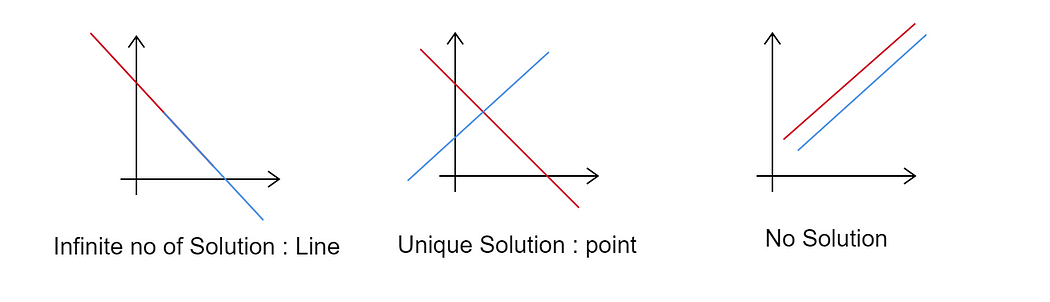
## Solution set of linear equations :

Every linear system may have only one of three possible number of solutions:

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

**Geometrical Representation :**

For a system of two variables (x and y), each linear equation determines a line on the xy-plane. The solution set is the intersection of these lines, and is hence either a line, a single point or don’t have any common point.



**Hyperplane**: This is a very important term in machine learning and is used very frequently. Below is a good explanation of hyperplane from wikipedia —

*A****hyperplane****is a subspace whose dimension is****one less than that of its ambient space****. If a space is 3-dimensional then its hyperplanes are the 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are the 1-dimensional lines. This notion can be used in any general space in which the concept of the dimension of a subspace is defined.*

Each linear equation determines a [hyperplane](https://en.wikipedia.org/wiki/Hyperplane) in n-dimensional space where n is the number of variables. The solution set is the intersection of these hyperplanes.

**Consistency :**A linear system is said to be consistent if it has at least one solution and is said to be inconsistent if it has no solution.

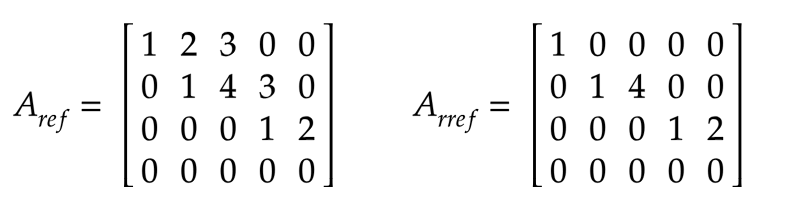
**Linear Independence :** A linear system is said to be independent if none of the equations can be written as a linear combination of others. For example equations x+y = 2 and 2x+2y = 4 are not linearly independent as the 2nd equation can be obtain by multiplying 2 with the 1st equation.

**Span:**The span of a set of vectors is the set of points obtained by all linear combination of the set of vectors. For example the vector set{(1,0,0), (0,1,0), (0,0,1)} span all the [real coordinate space](https://en.wikipedia.org/wiki/Real_coordinate_space) of 3 dimension (**R³).**Any n number of linearly independent vectors with real numbers can span a real coordinate space of dimension n. Span of a given set of vectors can be determined by the linearly independent vectors in the set.

**Rank of Matrix :**The maximum number of linearly independent rows of a matrix is called the **row rank**, and the maximum number of linearly independent columns is called the **column rank** of the matrix.

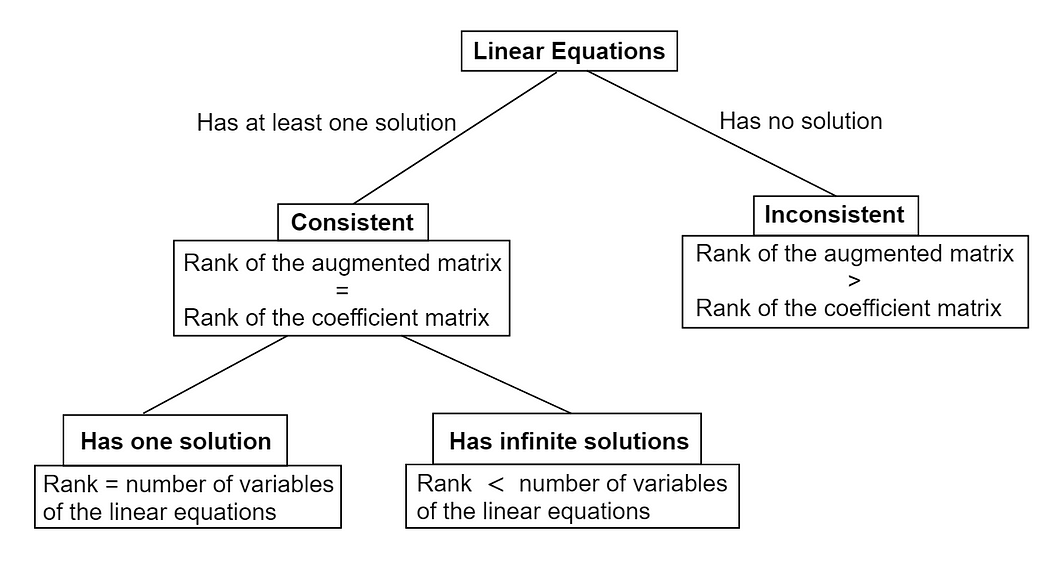
For any matrix A, **row rank of A = column rank of A = rank of A**

**How to determine rank of matrix:** Rank of matrix can be found by reducing the matrix in ‘**Echelon Form**’ s (Row Echelon Form or Reduced Row Echelon Form). This is done by elementary row operations and the whole method is called **Gaussian Elimination or Gauss–Jordan elimination.**



Row Echelon Form(ref) and Reduced Row Echelon Form (rref)

**Matrix Rank and number of solutions:**Now if we know the rank of **Coefficient**and **Augmented**matrices, then we can determine the number of solutions of a linear system of equations. Below is a diagram which will help you to understand the whole concept—



**Solution methods of linear equations:**There many different ways to solve a system of linear equation.

## How to Solve Linear Equations?

There are six main methods to solve linear equations. These methods for finding the solution of linear equations are:

* Graphical Method
* Elimination Method
* Substitution Method
* Cross Multiplication Method
* Matrix Method
* Determinants Method

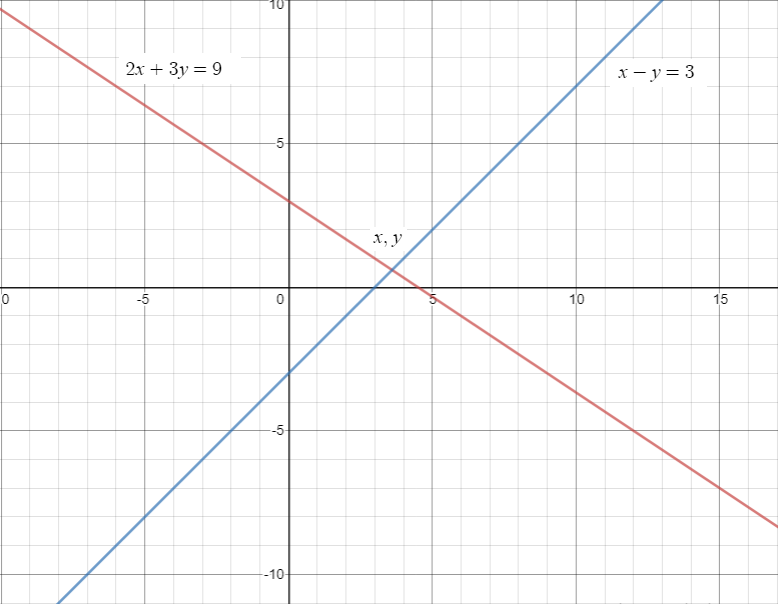
### Graphical Method of Solving Linear Equations

To solve linear equations graphically, first graph both equations in the same coordinate system and check for the intersection point in the graph. For example, take two equations as 2x + 3y = 9 and x – y = 3.

Now, to plot the graph, consider x = {0. 1, 2, 3, 4} and solve for y. Once (x, y) is obtained, plot the points on the graph. It should be noted that by having more values of x and y will make the graph more accurate.

**Graphical Method of Solving Linear Programming:**

The graph of 2x + 3y = 9 and x – y = 3 will be as follows:



In the graph, check for the intersection point of both the lines. Here, it is mentioned as (x, y). Check the value of that point and that will be the **solution of both the given equations**. Here, the value of (x, y) = (3.6, 0.6).

### Elimination Method of Solving Linear Equations

In the elimination method, any of the coefficients is first equated and eliminated. After elimination, the equations are solved to obtain the other equation. Below is an example of solving linear equations using the elimination method for better understanding.

Consider the same equations as

2x + 3y = 9 ———–(i)

And,

x – y = 3 ———–(ii)

Here, if equation (ii) is multiplied by 2, the coefficient of “x” will become the same and can be subtracted.

So, multiply equation (ii) × 2 and then subtract equation (i)

2x + 3y = 9

(-)

2x – 2y = 6

\_\_\_\_\_\_\_\_\_\_\_\_\_

-5y = -3

Or, y = ⅗ = 0.6

Now, put the value of y = 0.6 in equation (ii).

So, x – 0.6 = 3

Thus, x = 3.6

In this way, the value of x, y is found to be 3.6 and 0.6.

### Substitution Method of Solving Linear Equations

To solve a linear equation using the substitution method, first, isolate the value of one variable from any of the equations. Then, substitute the value of the isolated variable in the second equation and solve it. Take the same equations again for example.

Consider,

2x + 3y = 9 ———–(i)

And,

x – y = 3 ———–(ii)

Now, consider equation (ii) and isolate the variable “x”.

So, equation (ii) becomes,

x = 3 + y.

Now, substitute the value of x in equation (i). So, equation (i) will be-

2x + 3y = 9

⇒ 2(3 + y) + 3y = 9

⇒ 6 + 2y + 3y = 9

Or, y = ⅗ = 0.6

Now, substitute “y” value in equation (ii).

x – y =3

⇒ x = 3 + 0.6

Or, x = 3.6

Thus, (x, y) = (3.6, 0.6).

### Cross Multiplication Method of Solving Linear Equations

Linear equations can be easily solved using the cross multiplication method. In this method, the cross-multiplication technique is used to simplify the solution. For the cross-multiplication method for solving 2 variable equation, the formula used is:

**x /(b1 c2 − b2 c1) = y / (c1 a2 − c2 a1) = 1 /(b2 a1 − b1 a2)**

For example, consider the equations

2x + 3y = 9 ———–(i)

And,

x – y = 3 ———–(ii)

Here,

a1= 2, b1= 3, c1= -9

a2= 1, b2= -1, c2= -3

Now, solve using the aforementioned formula.

**x = (b1 c2 − b2 c1) / (b2 a1 − b1 a2)**

Putting the respective value we get,

x = 18/5 = 3.6

Similarly, solve for y.

**y = (c1 a2 − c2 a1) / (b2 a1 − b1 a2)**

So, y = ⅗ = 0.6

### Matrix Method of Solving Linear Equations

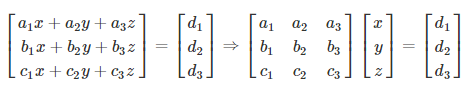
Linear equations can also be solved using matrix method. This method is extremely helpful for solving linear equations in two or three variables. Consider three equations as:

a1x + a2y + a3z = d1

b1x + b2y + b3z = d2

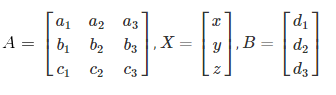
c1x + c2y + c3z = d3

These equations can be written as:



⇒ AX = B ————-(i)

Here, the A matrix, B matrix and X matrix are:



Now, multiply (i) by A-1 to get:

A−1AX = A−1B ⇒ I.X = A−1B

**⇒ X = A−1B**

### Determinant Method of Solving Linear Equations (Cramer’s Rule)

Determinants method can be used to solve linear equations in two or three variables easily. For two variables and three variables of linear equations, the procedure is as follows.

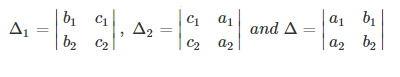
**For Linear Equations in Two Variables:**

x = Δ1/Δ,

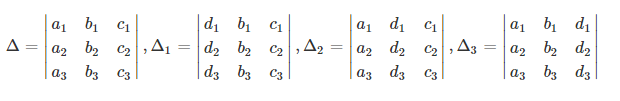
y = Δ2/Δ

Or, **x = (b1 c2 − b2 c1) / (b2 a1 − b1 a2)**and **y = (c1 a2 − c2 a1) / (b2 a1 − b1 a2)**

Here,



**For Linear Equations in Three Variables:**



### Methods of Solving Linear Equations in One Variable

Solving a linear equation with one variable is extremely easy and quick. To solve any two equations having only 1 variable, bring all the variable terms on one side and the constants on the other. The graphical method can also be used in which the point of intersection of the line with the x-axis or y-axis will give the solution of the equation.

For example, consider the equation 2x + 4 + 7 = 4x – 3 + x

Here, combine the “x” terms and bring them on one side.

So,

5x – 2x = 14

Or, x = 14/3

### Methods of Solving Linear Equations in Two Variables

To solve a linear equation in two variables, any of the above-mentioned methods can be used i.e. graphical method, elimination method, substitution method, cross multiplication method, matrix method, determinants method.

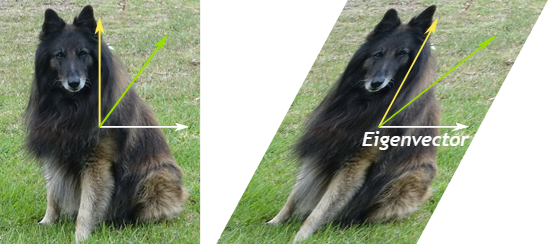
### Methods of Solving Linear Equations in Three or More Variables

For solving any equation having three or more variables, the graphical, elimination and the substitution method is not feasible. For solving a three-variable equation, the cross-multiplication method is the most preferred method. Even matrix Cramer’s rule is extremely useful for solving equations having 3 or more variables.

Eigen Values and Eigen vectors

A simple example is that an eigenvector **does not change direction** in a transformation: A simple example is that an eigenvector does not change direction in a transformation:

A simple example is that an eigenvector **does not change direction** in a transformation:



How do we find that vector?

**Eigenvalues** are associated with **eigenvectors** in Linear algebra. Both terms are used in the analysis of linear transformations. Eigenvalues are the special set of scalar values that is associated with the set of linear equations most probably in the matrix equations. The eigenvectors are also termed as characteristic roots. It is a non-zero vector that can be changed at most by its [scalar factor](https://byjus.com/maths/scale-factor/) after the application of linear transformations. And the corresponding factor which scales the eigenvectors is called an eigenvalue.

## Eigenvalue Definition

Eigenvalues are the special set of scalars associated with the system of linear equations. It is mostly used in matrix equations. ‘Eigen’ is a German word that means ‘proper’ or ‘characteristic’. Therefore, the term eigenvalue can be termed as characteristic value, characteristic root, proper values or latent roots as well. In simple words, the eigenvalue is a scalar that is used to transform the eigenvector. The basic equation is

**Ax = λx**

The number or scalar value “**λ”**is an eigenvalue of A.

## What are EigenVectors?

Eigenvectors are the vectors (non-zero) that do not change the direction when any linear transformation is applied. It changes by only a scalar factor. In a brief, we can say, if A is a linear transformation from a vector space V and **x** is a vector in V, which is not a zero vector, then v is an eigenvector of A if A(X) is a scalar multiple of **x**.

An **Eigenspace**of vector **x** consists of a set of all eigenvectors with the equivalent eigenvalue collectively with the zero vector. Though, the zero vector is not an eigenvector.

Let us say A is an “n × n” matrix and λ is an eigenvalue of matrix A, then **x**, a non-zero vector, is called as eigenvector if it satisfies the given below expression;

A**x** = λ**x**

**x** is an eigenvector of A corresponding to eigenvalue, λ.

**Note**:

* There could be infinitely many Eigenvectors, corresponding to one eigenvalue.
* For distinct eigenvalues, the eigenvectors are linearly dependent.
* Suppose, An×nis a square matrix, then [A- λI] is called an Eigen or characteristic matrix, which is an indefinite or undefined scalar. Where determinant of Eigen matrix can be written as, **|A- λI| and |A- λI| = 0** is the Eigen equation or characteristics equation, where “I” is the identity matrix. The roots of an Eigen matrix are called Eigen roots.
* Eigenvalues of a triangular matrix and diagonal matrix are equivalent to the elements on the principal diagonals. But eigenvalues of the scalar matrix are the scalar only.

# The four fundamental subspaces



The four fundamental subspaces of a matrix are the ranges and kernels of the linear maps defined by the matrix and its transpose. They are linked to each other by several interesting relations.

#### The Four Subspaces

Suppose that A is a m -by- n matrix that maps vectors in Rn to vectors in Rm . The four fundamental subspaces associated with A, two in Rn and two in Rm , are:

* row space of A, the set of all x for which Ax is nonzero,
* null space of A, the set of all x for which Ax=0=0,
* column space of A, the set of all y for which ATy is nonzero.
* left null space of A, the set of all y for which ATy=0

The row space and the null space are *orthogonal* to each other and span all of Rn. The column space and the left null space are also *orthogonal* to each other and span all of Rm.

The span of the rows of a matrix is called the row space of the matrix. The dimension of the row space is the rank of the matrix. The span of the columns of a matrix is called the range or the column space of the matrix. The row space and the column space always have the same dimension.

#### Dimension and rank.

The *dimension* of a subspace is the number of linearly independent vectors required to span that space. *The Fundamental Theorem of Linear Algebra* is

* The dimension of the row space is equal to the dimension of the column space.

In other words, the number of linearly independent rows is equal to the number of linearly independent columns.

The *rank* of a matrix is this number of linearly independent rows or columns.

## **Column Space**

The **column space** of a matrix *A* is the vector space formed by the columns of *A*, essentially meaning all linear combinations of the columns of *A*. Equivalently, the column space consists of all matrices *Ax* for some vector *x*.

For this reason, the column space is also known as the **image** of *A* ((denoted ,im(*A*)), as it is the result when *A* is viewed as a linear transformation of the vector space R*m* (where *m* is the number of rows of *A*). In particular, the image of *A* is necessarily a subspace of R*m*, hence the term "fundamental subspace."

## **Nullspace**

The **nullspace** or **kernel** of a matrix *A* (ker(*A*)) is the set of all vectors *x* for which *Ax*=0

### Applications

* **Calculation of Pseudo-inverse:**Pseudo inverse or Moore-Penrose inverse is the generalization of the matrix inverse that may not be invertible (such as low-rank matrices). If the matrix is invertible then its inverse will be equal to Pseudo inverse but pseudo inverse exists for the matrix that is not invertible. It is denoted by A+.

