What is Vector Algebra?

- What is a Vector?
- What is Algebra?
- Symbolic Computing

What is a Vector?

Any set of objects with addition, subtraction and multiplication is an algebra. Any such object requiring a **fixed number of symbols** to describe itself is called a vector. These are informal definitions. A collection of such vectors with operations such as **+** or **-** is called a Vector Algebra.



Here: is the start point or the **pivot**.

There: is the finish point or the **endpoint**.

Select any two symbols and you can create a tuple namely the **left CurlyBracket** '**{**' and the **right Curly Bracket** '**}**' for example:

```
{a,b} or {c,d}
```

Now let's **Add** these two vectors:

```
{a,b} + {c,d} = {a + c,b + d};
```

Operator: any symbol in mathematics, that indicates an **operation** to be performed e.g. +

'**+**' : **Addition Operator** of the form ◧ op ◨

◧ + ◨

◧: **Left Operand** of an Operator

◨: **Right Operand** of an Operator

'**=**': Identity Operator of the form ◧ op ◨

'**,**' : **Comma** or **Separator Operator** of the form ◧ op ◨

'**{ }**': **Brackets** or **Grouping** or **Bracket Operator**, grouping the enclosed circle between the right and left parenthesis

What is Algebra?

> **What**: Concept of Algebra
> **Why**: What is an algebra why you need algebra
> **Time To Complete**: 1 hour



Nature of Algebra

Algebra is a set of finitely many symbols with functions and operators acting on these symbols in such a fashion that all these functions and operators can be implemented in an electromechanical device or a computer or computer program.

A human being or a machine, by applying mechanically oriented operations could simply operate on the symbols of an algebra without any requirements to understand what the symbols or the operations are supposed to mean!

The latter is called **Abstraction**.

Example: The *Comma operator*

a , b

separates the ‘a’ from ‘b’ so it is not confused with ‘ab’ or in general separates the Right Hand Side (rhs) from Left Hand Side (lhs).

Example: The *Function* Sin

Sin(a)

computes the well known trigonometric function Sinus.

Words

A Word in an algebra is a sequence of concatenated symbols and functions and operators.

Example:

c = a + b

symbols c, =, a, +, and b are concatenated or sequenced into a string of letters or a word as the final ensemble.

Valid Words

Not all sequences of symbols in an algebra are Valid, indeed there is another ‘mechanical’ process of an algebra which qualifies a word either being Valid or Invalid.

Example:

a = b + 1 a valid word in algebra of arithmetic

1 +++, b an invalid word in algebra of arithmetic

Repeatability

The first motivation and the last goal of an algebra is one and only one namely **Repeatability**:

> If a set of symbols and operations manipulated by a person living in the 17th century was again manipulated by a person today, both efforts yield the same results!

This is due to the fact that Algebra at its core construction is like unto an electromechanical entity that requires no intelligence to operate! Even a machine can carry out the same operations!

> Students need to be aware that being good at algebra or making proficient use of algebraic systems does not require any intelligence, rather requires skills and in particular skills in programming. A person being good at algebra is decided by their skills to manipulate symbols and with no reference to his/her level of intelligence.

Symbolic Computing

> **What**: Concept of Symbolic Computing
> **Why**: Computing with symbols and non-numerals
> **Time To Complete**: 1 hour



Why?

As discussed in previous class notes, the nature of algebra is conducive to expand our abilities to understand and compute geometries.

Symbolic Computing, in part, is the very software engine that functions the said algebra's symbols and expressions.

Therefore, naturally Symbolic computing goes hand in glove with algebra.

What?

See simple example of use of Free Form language's symbolic computing capabilities. Notice no numbers, all in symbols.

```
var1 = a + b;
var2 = a - b;

res = var1 * var2;

show res;

res2 = expand[var1 * var2];

show res2;

save as symbolics;
```

Output:

```
{
res  =  (a - b)*(a + b)
,
res2  =  a^2 - b^2
}
```