# **Continuous Function**

A continuous function is a function that has no abrupt changes in value, known as discontinuities. A small change in the input of a continuous function produces only a small change in the output. A continuous function can be drawn without lifting the pencil. A formal way to define a continuous function is using the concept of open sets. [A function is continuous if the pre-image of every open set in the output is open in the input](https://www.dictionary.com/browse/continuous-function).

Some examples of continuous functions are polynomial functions, exponential functions, trigonometric functions, logarithmic functions, and absolute value functions. Some functions are continuous only on certain intervals or domains, such as square root functions, rational functions, and inverse trigonometric functions. Some functions are discontinuous at certain points, such as piecewise functions, step functions, and Dirichlet functions.

Definition of a continuous function

∀ϵ ∈ ℝ+: ∃δ > 0 s.t.

∥x − y∥2 < δ ⇒ ∥f(x) − f(y)∥2 < ϵ

“For all epsilon belonging to the positive real numbers, there exists a delta greater than zero such that the norm of x minus y is less than delta implies that the norm of f(x) minus f(y) is less than epsilon.”

It can be explained as follows:

* A function f is continuous at a point x = y if for any positive number ϵ (epsilon), there exists another positive number δ (delta) such that the distance between f(x) and f(y) is less than ϵ whenever the distance between x and y is less than δ.
* In other words, no matter how close we want the output values of the function to be, we can always find a range of input values around y that will guarantee that closeness.
* This means that the function has no gaps, jumps, or breaks at x = y, and its graph can be drawn without lifting the pen.

# **Scripts are Interpreted by a REPL Program**

REPL (Read-eval-print loop) is a term that refers to a simple interactive computer programming environment that takes single user inputs, executes them, and returns the result to the user. A program written in a REPL environment is executed piecewise, meaning that the user can enter one or more expressions and see the output immediately, without having to compile or run the whole program. REPLs are useful for exploratory programming, debugging, and learning new languages or tools.

Some examples of REPLs are:

* [The Node.js console, which allows you to run JavaScript code in a terminal or a web browser](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop)
* [IPython, which is an enhanced Python shell that supports interactive data analysis, visualization, and parallel computing](https://www.digitalocean.com/community/tutorials/what-is-repl)
* [The Bash shell, which is a command-line interface for Unix-like operating systems that lets you run commands and scripts](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop)
* The developer console found in most web browsers that lets you inspect and modify the web page elements, run JavaScript code, and access various web development tools.

# **Compiled Languages Vs Interpreted Languages**

Compiled languages and interpreted languages are two types of programming languages that differ in how they are executed by a computer. A compiled language is a language that is translated into machine code, which is the binary code that the computer can directly understand and execute. An interpreted language is a language that is not translated into machine code, but rather executed by another program called an interpreter, which reads and executes the source code line by line.

Some of the advantages of compiled languages are:

* They are faster and more efficient since they do not need an interpreter to run.
* They give more control over the hardware aspects, such as memory management and CPU usage.
* They are more secure since the source code is not exposed to the user or other programs.

Some of the disadvantages of compiled languages are:

* They need a separate compilation step before running, which can take time and resources.
* They are less portable since the machine code is specific to the target platform and architecture.
* They are harder to debug since the errors are detected only at compile time and not at run time.

Some of the advantages of interpreted languages are:

* They are easier to learn and use, since they have simpler syntax and features.
* They are more flexible, since they can be modified and executed on the fly without recompiling.
* They are more portable, since they can run on any platform that has an interpreter for that language.

Some of the disadvantages of interpreted languages are:

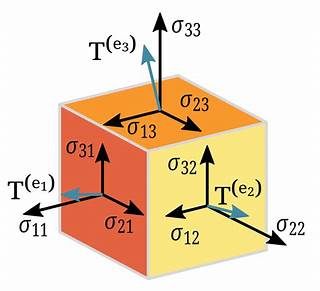
* They are slower and less efficient since they need an interpreter to run and interpret each line of code.
* They have less control over the hardware aspects, such as memory management and CPU usage.
* They are less secure since the source code is exposed to the user or other programs.

Some examples of compiled languages are C, C++, C#, Rust, and Go. Some examples of interpreted languages are Python, Ruby, JavaScript, and PHP. However, some languages can have both compiled and interpreted implementations, depending on the specific needs and preferences of the developers.

Some examples of languages that can be both compiled and interpreted are:

* **Ada**: a multi-purpose language that was originally designed for embedded and real-time systems, but has also been used for high-level applications, such as aviation and banking. [Ada can be compiled to native code or interpreted by a virtual machine1](https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/)
* **BASIC**: one of the oldest and most widely used programming languages, especially for beginners and education. [BASIC can be compiled to executable files or interpreted by various implementations, such as Visual Basic and QBasic1](https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/)
* **Java**: a popular object-oriented language that is widely used for web, mobile, and desktop applications. [Java can be compiled to bytecode, which is then executed by a virtual machine, or interpreted by a just-in-time compiler, which converts bytecode to native code on the fly](https://bing.com/search?q=languages+that+can+be+both+compiled+and+interpreted)
* **Python**: a high-level, general-purpose language that emphasizes readability and simplicity. [Python can be compiled to bytecode, which is then executed by a virtual machine, or interpreted by various implementations, such as CPython and PyPy](https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/)

**Tensors**



Tensors are mathematical objects that can be used to describe various physical phenomena, such as forces, stresses, strains, electric and magnetic fields, and more. They are generalizations of scalars, vectors, and matrices, which have no, one, or two indices, respectively. Tensors can have any number of indices, depending on the dimension of the space they are defined on. For example, a tensor with three indices can be represented by a three-dimensional array of numbers, or a cube of numbers.

Tensors are defined independently of any coordinate system or basis, which means that they have the same meaning regardless of how we choose to measure or represent them. However, to perform calculations with tensors, we need to choose a basis and express the components of the tensor in that basis. Different bases may give different components for the same tensor, but the tensor itself does not change. To transform the components of a tensor from one basis to another, we use special rules that depend on the type and rank of the tensor.

Tensors are useful in physics because they allow us to express physical laws in a way that is invariant under changes of coordinates. For example, the stress tensor is a second-order tensor that describes the force per unit area acting on a material at a given point. The stress tensor does not depend on the orientation of the material or the observer, but only on the intrinsic properties of the material and the external forces applied to it. The stress tensor can be used to study the deformation and fracture of materials under various conditions.

**Norms**

A norm in linear algebra is a function that assigns a positive number to any vector, representing its length or magnitude. A norm must satisfy four properties:

* **Non-negativity**: The norm of any vector is always greater than or equal to zero, and it is zero only for the zero vector.

That is, ∥v∥ ≥ 0 for all v, and ∥v∥ = 0 if and only if v = 0.

* **Absolute homogeneity**: The norm of a scaled vector is equal to the absolute value of the scalar times the norm of the original vector.

That is, ∥αv∥ = |α|∥v∥ for any scalar α and any vector v.

* **Triangle inequality**: The norm of the sum of two vectors is less than or equal to the sum of their norms.

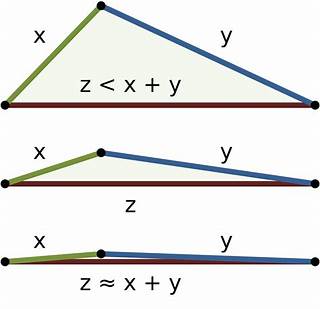
That is, ∥u + v∥ ≤ ∥u∥ + ∥v∥ for any vectors u and v.

* **Subadditivity**: The norm of the difference of two vectors is greater than or equal to the difference of their norms.

That is, ∥u - v∥ ≥ |∥u∥ - ∥v∥| for any vectors u and v.

These properties ensure that a norm behaves like a measure of distance or size, and they are useful for studying various aspects of linear algebra, such as convergence, orthogonality, and linear transformations. There are different ways to define a norm on a vector space, such as the Euclidean norm, the Manhattan norm, the infinity norm, and so on. Each norm has its own advantages and disadvantages, depending on the context and the application. [You can find more information and examples about norms and their properties in the web search results I found for you1](https://medium.com/linear-algebra/part-18-norms-30a8b3739bb)[2](https://bing.com/search?q=properties+of+any+norm+in+linear+algebra)[3](https://en.wikipedia.org/wiki/Norm_%28mathematics%29)[4](https://datacadamia.com/linear_algebra/norm)

**Triangle Inequality**



“Triangle inequality” is a term that refers to a theorem or a property in mathematics that relates the lengths of the sides of a triangle. It states that the sum of the lengths of any two sides of a triangle is always greater than or equal to the length of the third side. For example, if a, b, and c are the lengths of the sides of a triangle, then the triangle inequality is written as:

a + b ≥ c b + c ≥ a and a + c ≥ b

The triangle inequality can be understood intuitively by imagining that the shortest distance between two points is a straight line. Therefore, if we have three points that form a triangle, the length of any side of the triangle must be shorter than or equal to the length of the path that goes through the other two points. The equality holds only when the three points are collinear, which means that the triangle has zero area and is degenerate.

The triangle inequality is not only a theorem in Euclidean geometry, but also a defining property of norms and measures of distance in other types of geometry and spaces. A norm is a function that assigns a positive number to any vector or array, representing its magnitude or size. A measure of distance is a function that assigns a positive number to any pair of points, representing their separation or difference. For any norm or measure of distance, the triangle inequality must hold, meaning that the norm or distance of the sum of two vectors or points is less than or equal to the sum of their norms or distances.

The triangle inequality has many applications and generalizations in mathematics and physics, such as the law of cosines, the Pythagorean theorem, the Cauchy-Schwarz inequality, the Hahn-Banach theorem, and the Heine-Borel theorem. [You can find more information and examples about the triangle inequality and its related topics in the web search results I found for you](https://en.wikipedia.org/wiki/Triangle_inequality)[1](https://en.wikipedia.org/wiki/Triangle_inequality)[2](https://mathworld.wolfram.com/TriangleInequality.html)[3](https://brilliant.org/wiki/triangle-inequality/)[4](https://www.britannica.com/science/triangle-inequality)

# **Probability**

**Probability space** is a way of modeling a random experiment or process using mathematics. It consists of three parts:

* **A sample space**, which is the set of all possible outcomes of the experiment. For example, if you toss a coin, the sample space is {heads, tails}.
* **An event space**, which is a collection of subsets of the sample space that we are interested in. These subsets are called events, and they represent possible outcomes or combinations of outcomes. For example, if you toss a coin twice, an event could be {two heads}, {one head and one tail}, or {at least one head}.
* **A probability function**, which assigns a number between 0 and 1 to each event, indicating how likely it is to occur. The probability function must follow some rules, such as the probability of the whole sample space is 1, the probability of the empty set is 0, and the probability of the union of disjoint events is the sum of their probabilities.

A probability space allows us to calculate the probabilities of various events and compare them.

# **The Difference Between a Sample Space and an Event Space**

The difference between a sample space and an event space in a toss of a coin is that:

* A sample space is the set of all possible outcomes of the experiment. For a coin toss, the sample space is {h, t}, where h means heads and t means tails.
* An event space is a collection of subsets of the sample space that we are interested in. These subsets are called events, and they represent possible outcomes or combinations of outcomes. For example, an event could be {h}, {t}, or {h, t}.
* An event space must include the empty set, which represents no outcome, and the sample space itself, which represents any outcome. It must also satisfy some other properties, such as being closed under union, intersection, and complementation.

So, is it true to say the event space is the power set of the sample set?

It is not always true to say that the event space is the power set of the sample set. It depends on the nature of the sample set and the probability measure. If the sample set is finite or countable, then the event space can be the power set of the sample set, and every subset of the sample set can be assigned a probability. However, if the sample set is uncountable, such as the real numbers, then the event space cannot be the power set of the sample set, because some subsets of the sample set are not measurable, meaning that they cannot be assigned a probability in a consistent way. In this case, the event space is a smaller collection of subsets of the sample set, called a sigma-algebra, that satisfies some properties that make it suitable for probability theory.

# **The support of a random variable**

The support of a random variable is the set of values that the random variable can take with non-zero probability. For example, if you roll a fair six-sided die, the support of the random variable that represents the outcome is {1, 2, 3, 4, 5, 6}, because each of these values has a positive probability of occurring. The support of a random variable depends on its probability distribution, which can be discrete or continuous. For discrete random variables, the support is the set of all possible values that the random variable can take. For continuous random variables, the support is the set of all real numbers where the probability density function is positive. You can learn more about the support of a random variable from the following web pages: [Support of a random variable - Statlect](https://www.statlect.com/glossary/support-of-a-random-variable), [Support of a random variable - Mathematics Stack Exchange](https://math.stackexchange.com/questions/846011/precise-definition-of-the-support-of-a-random-variable), and [Support vs range of a random variable - Mathematics Stack Exchange](https://math.stackexchange.com/questions/416035/support-vs-range-of-a-random-variable).