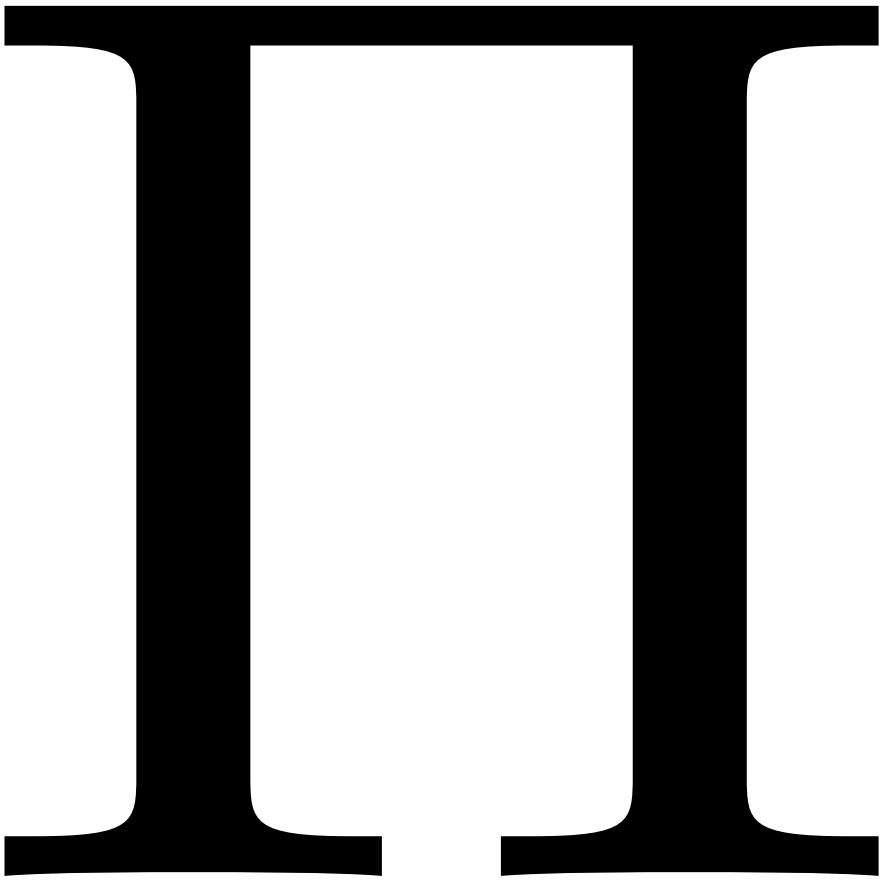
Mathos Project

Exploring the computational power for mathematical calculations

---

By Mathos Project Community



Contents

Part

Mathos Core Library

1 | Uncertainty Number

Mathos Core Library Uncertain Number class has undergone some changes, one of them is being able to work with custom functions. In this article, we are to focus on this new feature mainly.

The ability to use custom functions is split up into four methods with the name *CustomFunction* but with different input parameters and different modifiers. A list of this function with different method signatures is shown below:

```
1 public UncertainNumber CustomFunction(Func<decimal, decimal>
      function)
2
3 public static UncertainNumber CustomFunction(Func<decimal[],
      decimal> function, params UncertainNumber[] points )
4
5 public static UncertainNumber[] CustomFunction(Func<decimal,
      decimal> function, string TSV)
6
7 public static UncertainNumber[] CustomFunction(Func<decimal[],
      decimal> function, string TSV)
```

1.1 Basics

As you can see, all of them take a function that can be customized in any way using for instance a lambda expression. Let's look at an example of the first function in the list. Say we have declared an Uncertain Number with a value and an uncertainty and want to perform a mathematical calculation with the value, and at the same time keep its uncertainty. As an example, the function $y = 5a^2$ is going to be used.

```
1 // remember to include Mathos.Statistics.Uncertainty;
2
3 UncertainNumber x = new UncertainNumber(32.7M, 2);
4 x = x.CustomFunction(a => 5 * a * a);
5
6 x = x.AutoFormat();
7
8 Console.WriteLine(x);
```

This will return 5300 ± 700. Notice that *AutoFormat* makes the value and the uncertainty look in the way they should when you display this data in a rapport.

1.2 More variables

The other three options in the list are static, therefore, in contrast to the first method (described in Basics), you can access them without creating a new instance of the UncertainNumber class.

Let's look at the second method. That will allow us to have more than one variable inside the function. For this example, we are to use three variables.

```
1 UncertainNumber result = UncertainNumber.CustomFunction(a => a
      [0] * a[1],
2 new UncertainNumber(2, 0.1M), new UncertainNumber(3, 0.1M));
3 result = result.AutoFormat();
4
5 Console.WriteLine(result);
```

This will write 6.0 ± 0.5 on the screen. The function that was used in the calculation is $f(x, y) = xy$. Notice that `a[0]` is the first variable and corresponds to the first element of the UncertainNumber array, which in this case is 2.0 ± 0.1.

1.3 Working with raw data

The last two functions allow you to include a TSV (tab separated values) string into the method. This means that you can copy a table from for example Microsoft Excel and get the uncertainties calculated for you. The last function allows you to use more than one variable, while the second last is restricted to one variable only. Both these functions are static, and so have to be included directly through UncertainNumber class (you don't need a new instance of that class). Note that these two functions return an array of UncertainNumber.