Extrascript documentation

This document is written in order to clarify certain aspects of the script which couldn't be clarified on-spot due to its length. It's written a-la ISO 9001, so its structure should be accessible for everyone.

Context:

The program is written in order to find the n-th root of a certain number using the bisection algorithm. In order to make it faster, it also searches for former results just in case we already have that result in our hands

Vocabulary:

n-th root: A certain (n) root of a number. Example gratia, $4^{1/5}$ is the fifth root of 4.

Base: The number we are trying to find the n-th root of. In our $4^{1/5}$ example, 4 is the base.

Sensibility: The accuracy we want for our result. It's measured in positive integers, and it's expressed as a negative power of 10, which we call quasizero. If our sensibility is 4, it's actually $10^{-4} = 0.0001$. The higher the number, the closer it's to zero thus the more accurate it is.

Normalization/denormalization: Process made in order to assimilate/portray the information given the kind of root. As we are searching on the positive real numbers we have to both find the negative root in even roots (which is the same as the positive one except it's on the mirrored position) and the uneven root of a negative number (which is a negative number which is the opposite of the same root of the number opposite to the base one). Any other case doesn't go through this.

Extrascript documentation

Normalization kind/type: the normalization/denormalization path we have to follow. It's "even" if it's an even root, "uneven-negative" if it's an uneven root of a negative number, and "normal" in any other case.

Target function: mathemathical function which we use to define our problem.

Theorical basis:

Bolzano's Theorem:

Given a certain function which is continuous in a certain interval [a, b] if $f(a)\cdot f(b) < 0$, there is a c in [a, b] where $f(c) = 0$.

Bisection algorithm:

Given a certain function f(x), if it's continuous in an interval [a,b] where we know there is at least a number (s) which fulfills that $f(s) = 0$, for finding s we split [a, b] in half with the middle point being $m = (a + b)/2$.

This way we have three options, either $f(m) = 0$, either $f(m)\cdot f(a) < 0$, either $f(m)\cdot f(b)$, which we shall have any combination of the three with at least one option. If the second or third options are fulfilled, we shall iterate over [a, m] and [m,b] respectively up until we find s.

Applied bisection algorithm:

It's the same as before but our s doesn't equal zero but quasizero (qzero in the code). Our target function is $t(x) = b – x^r$, where b is the base and r the root.

First, we normalize the root as stated beforehand. Then we set the middle point, then we check this with the target function. Being our interval [a, b] and the middlepoint m, we have three mutually exclusive options in positive real numbers, either $t(m) < qzero$, either $t(m)\cdot t(a) < 0$ and our solution is in [a, m], either $t(m)\cdot t(b) < 0$ and our solution is in [m, b]. In the first case, we have found our number and we stop. On the second one, we shall iterate this on [a, m] and on the third one on [m, b]

Extrascript documentation

Conclussion:

As per the class is defined today it’s 100% functional and as optimized as it’s within my limits. Also, it’s a clear example on how to apply the bisection algorithm as intended.