Bezier Curves

Octave Project 3

# Description of the problem:

The mathematical problem we are trying to solve with these methods is interpolating 2D or 3D curves using a set of control points. Bezier curves are defined as a linear combination of Bernstein polynomials as follows. At the end we are using those control points to create the Bernstein polynomials that define the curve.

# Explanation of the method:

**Direct Evaluation:**

**De Casteljau:**

Given P0, P1 … Pn points in or the Bezier curve can be computed constructing a regular mesh of m + 1 nodes in [0, 1] and then applying linear interpolation recursively on for each node. The recursion is noted as follows:

Texto, Carta

Descripción generada automáticamenteAnd in code it looks like this

**Midpoint:**

Given P0, P1 … Pn points in or the Bezier curve can be split into two different curves, starting at P0  and ending at where , hence the name of midpoint subdivision, and starting at  and ending at Pn. Using De Casteljau evaluating at  we can compute the first midpoint:

From this we can use , , , as the new control points to construct the curve , which can be defined by the following formula:

Similarly, can also be constructed by using , , , as control points, which can be defined by the following formula:

Furthermore, we can keep subdividing the curves and to approximate our curves with more precision and smoothness. Code implementation wise this algorithm has exponential growth which is something to avoid, that is why when implemented usually is restricted to a certain amount of iterations for the subdivision.

# Examples:

**Direct Evaluation:**

**De Casteljau:**

Given .

For :

For :

Gráfico, Gráfico de líneas

Descripción generada automáticamenteFor :

Gráfico, Gráfico de líneas

Descripción generada automáticamenteOutput of the program compared with the one obtained by hand.

Midpoint Subdivision:

# Bibliography:

MAT300 Lecture Notes: lecture11. MAT300 Lecture Notes: lecture12.