A Panorama of Geometric Modeling

Jean Gallier

1. What is Computer-Aided Geometric Design?

- CAGD = Techniques for modeling (complex) curved shapes, using curves and surfaces.
- Such techniques are used in 2D and 3D drawing and plot
- product design and manufacturing (cars, planes, ships. etc.)
- architecture (buildings)
- computer animation (movies, computer simulation)

CAGD

- medical imaging: modeling of organs (brain, prostate, liver, etc.)
- topographic data, weather data,
- robotics: motion interpolation
- computer vision: 3D models from "point clouds"
- biology and computational chemistry
- archeology, art!

Some Historical Backgroud

Even in the nineteenth century, ship hulls were designed using long flexible wooden strips called splines.

In the fourties, engineers at BOEING experimented with computer-aided methods for designing airplanes.

Various types of surfaces amenable to efficient computer processing were invented.

In the 1960's, independently, two French engineers working for Citroen and Renault pioneered the use of CAGD in car design.

Practical Uses in Industry

Subdivision surfaces are used by

- Intel: scalable 3D models (see home page)
- Pixar: Entertainmemt industry, computer animated movies. (show movie “Geri’s game”).

3-D modeling, simulation and virtual reality tools, used by BOEING and other aerospace companies (see phantom work).

More Historical Backgroud

Both Pierre Bézier and Paul de Casteljau proposed schemes for specifying curves and surfaces using control points, as opposed to explicit coefficients.

In the early 1970's, Carl de Boor (U. of Wisconsin) and Richard Riesenfeld (U. of Utah) came up with B-splines curves and surfaces and the "de Boor algorithm".

In 1978, Catmull and Clark proposed subdivision surfaces. Catmull is now one of the vice presidents of Pixar!

2. Problems Considered

(1) Approximating a curved shape. In this type of problem, accuracy is not so important but smoothness usually is. “Free-hand drawing” is a 2D analog.

(2) Interpolating a curved shape. Here, some data points are given and the interpolating curve or surface must pass through some required points.

(3) Rendering (i.e., drawing) smooth curves or surfaces.

Example: Interpolation of Data Points Using a Curve

Problem: Given data points $x_0, \cdots, x_n$ (in the plane), find a curve starting at x_0, ending at x_n, and passing through al the points.

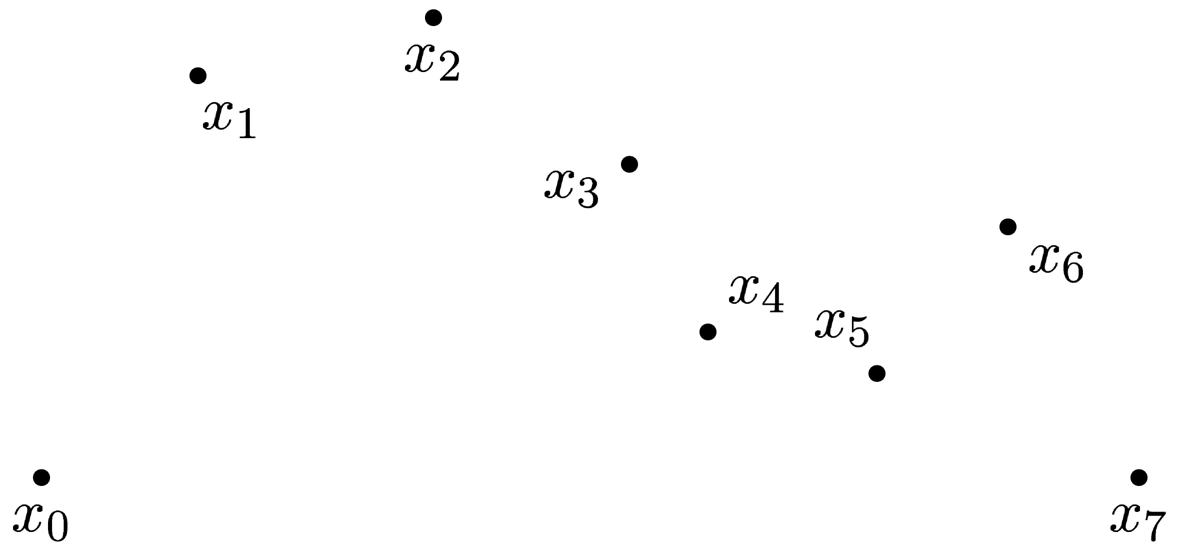

Figure 1: Data points $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$

What's a good interpolation curve?

Clearly, there are infinitely many curves interpolating the data points; how do we choose a "good" curve.

We need some criteria for what is a good curve.

For example

(a) A smooth curve (what does this mean?)

(b) A curve that does not wiggle excessively.

(c) A curve that can be easily modified locally, without affecting the entire curve.

Interpolation Using a Cubic Spline Curve

If we use Bézier spline curves of degree m (usually, $m = 3$), then we get good smoothness and no excessive wiggling.

Incremental changes are easily accomodated (it's called local control).

Curves are unique up to the specification of end conditions (for example, prescribed tangent vectors).

There are efficient algorithms to draw such curves.

Closed curves can also be handled.

A cubic interpolation spline curve

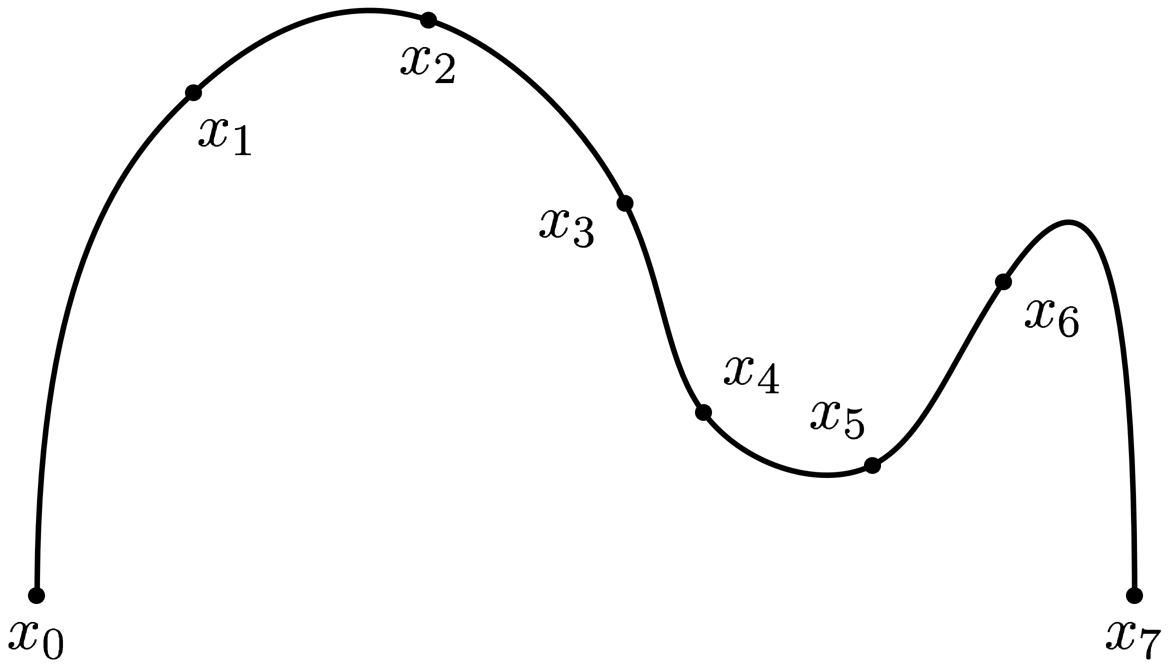

Figure 2: A cubic interpolation spline curve passing through the points $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$

How is a spline specified?

Spline curves consist of little pieces of (polynomial) curves glued together.

All individual curves have the same degree (often $3 =$ cubic curves).

The spline curve is specified by a sequence of line segments, its control polygon; the vertices are the control points.

There is an algorithmic method for drawing the spline curve from its control polygon.

Note: The control points usually differ from the data points.

Control polygon of a spline

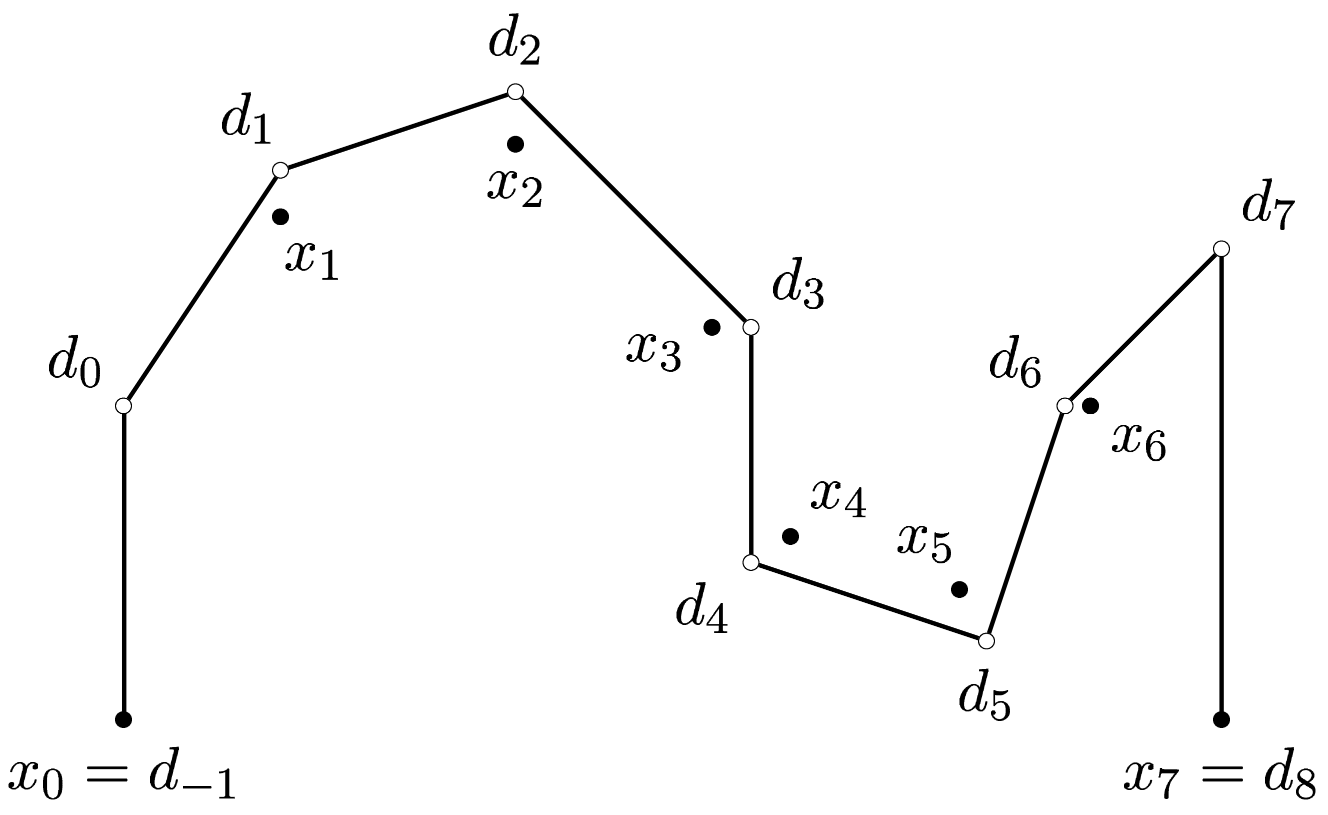

Figure 3: Control polygon of a cubic spline through $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$

A cubic spline and its control polygon

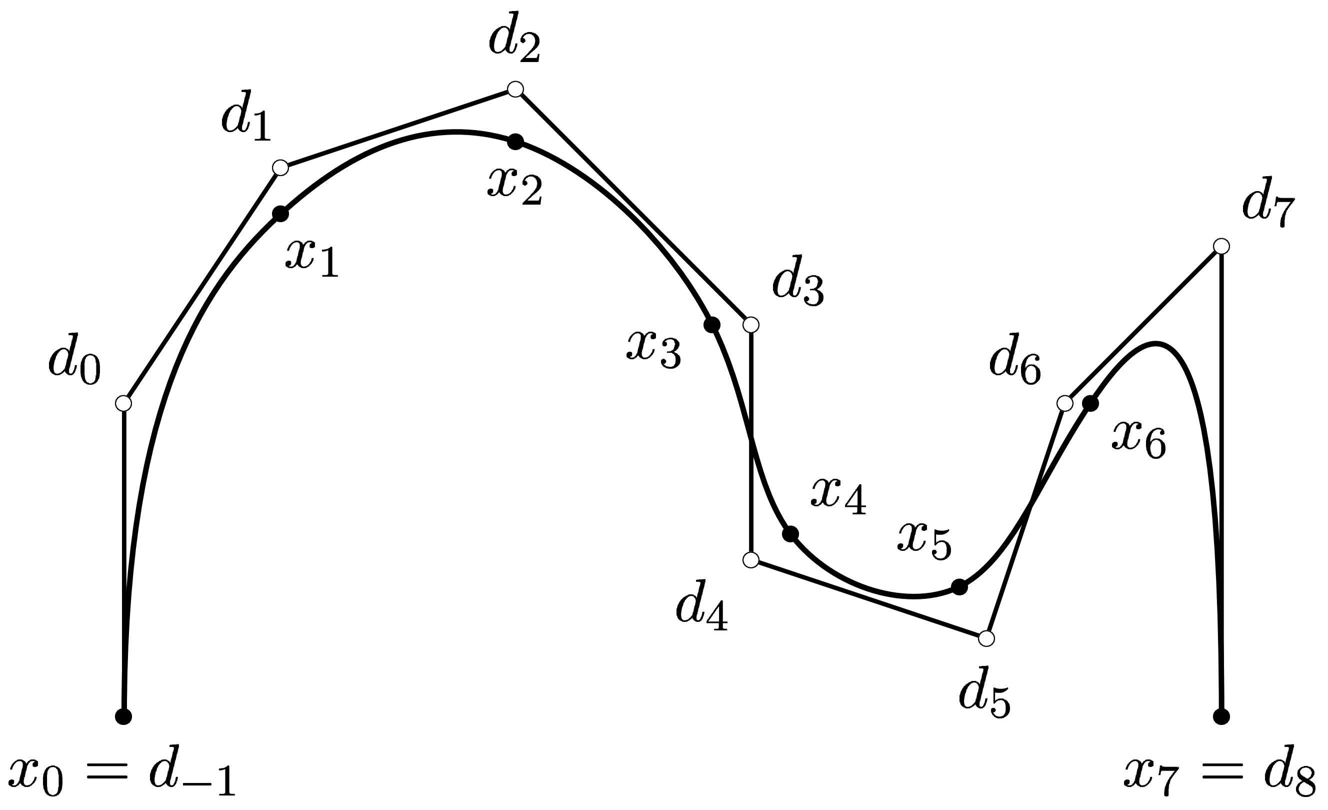

Figure 4: A cubic spline curve through $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$ and its control polygon

3. What is a Curve?

Traditionally, there are two ways of defining curves:

(1) Implicitly: The curve is the zero-locus of one or more equations. For example, consider the unit circle:

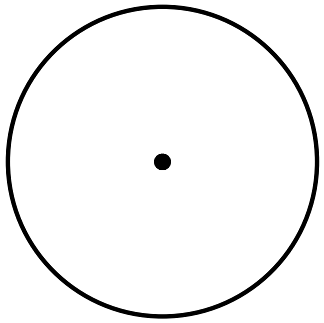

Figure 5: A unit circle

This circle is defined implicity by

$$x^2 + y^2 = 1.$$

What is a Curve?

(2) Parametrically: The curve is given as the set of points (x, y) such that $x = f(t), y = g(t)$. For example, the same circle is given by

$$x = \cos t, \quad y = \sin t.$$

Another (cheaper!) rational parametrization:

$$x = \frac{1 - t^2}{1 + t^2}, \quad y = \frac{2t}{1 + t^2}.$$

(How do we get the point $(-1, 0)$?)

Implicit versus Parametric

Implicit definitions are good if we want to decide whether or not a point belongs to a curve, but not good to draw the curve.

Parametric definitions are much better to draw a curve, but what kind of functions should we allows for f, g?

We would like functions that are "cheap" to compute.

Polynomials (or ratios of polynomials) are cheap to evaluate.

Polynomial and Non-Polynomial Functions

The function $f(t)$ given by

$$f(t) = 2t^3 - t^2 + 3t - 1$$

is a polynomial (of degree 3).

The function

$$f(t) = e^{-t^2} \cos t + t \sin t$$

is not a polynomial. The function

$$f(t) = \frac{1 - t^2}{1 + t^2}$$

is not a polynomial, but it is the next best thing: a rational function.

The idea of a spline

But, polynomial curves of high degree are hard to control. They wiggle too much!

It is may also be too expansive to use polynomials of high degree. It will take too long to draw such curves and there will be precision issues.

Instead of using a single polynomial curve use a piecewise polynomial curve, that is, break up a complicated curve into small manageable simple curves of low degree (usually 3). Such curves are spline curves.

Dealing with each piece is easy and efficient. There are good approximations algorithms, for example, the de Casteljau algorithm (see below).

Joining the pieces properly

The difficulty in dealing with splines is not to handle the small pieces constituting a spline, but to insure that these pieces join with an appropriate amount of smoothness.

Sometimes, it is desirable to have various degrees of smoothness.

This is technically complicated, but good control of smoothness can be achieved.

A Spline with Varying smoothness

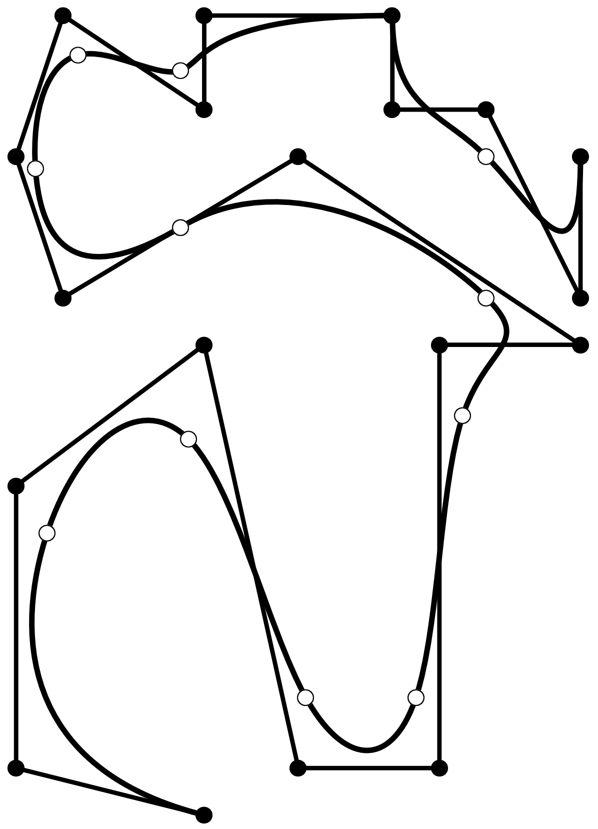

Figure 6: A cubic spline with varying smoothness

4. Bézier Curves

We now take a closer look at the individual polynomial curves of low degree that form a spline.

What's great is that it is possible to define such curves in terms of control points rather than in terms of explicit polynomials.

Furthermore, this approach leads to efficient recursive algorithms for drawing such curves, with any prescribed degree of accuracy.

This method is due to Bézier, but the theory was know to nineteenth century mathematicians. But they did not have the practical motivations!

Control Points for Polynomial Curves

If a curve, C, is given by

$$x = f(t), \quad y = g(t),$$

where $f(t)$ and $g(t)$ are polynomials of degree at most m, we say that C is a curve of degree m.

In general, a polynomial curve of degree m is specified by $m+1$ control points, $b_0, \ldots, b_m$.

So, a straight line is given by two points (surprise!).

Control Points for Polynomial Curves

A curve of degree 2 is given by 3 points

A curve of degree 3 is given by 4 points, etc.

Here is an example of a control polygon when $m = 3$:

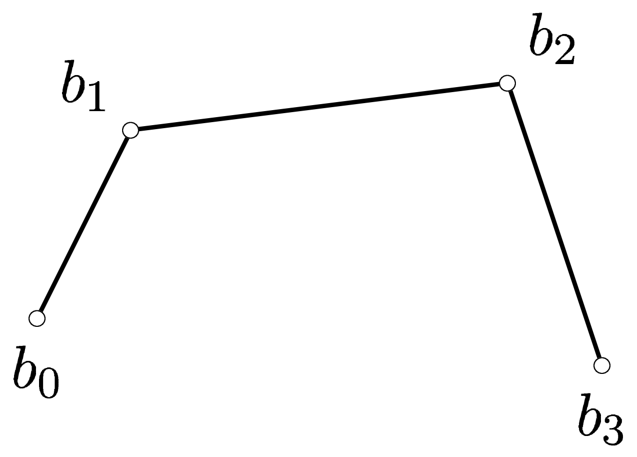

Figure 7: Control points and polygon for a cubic curve

A Curve and its control polygon

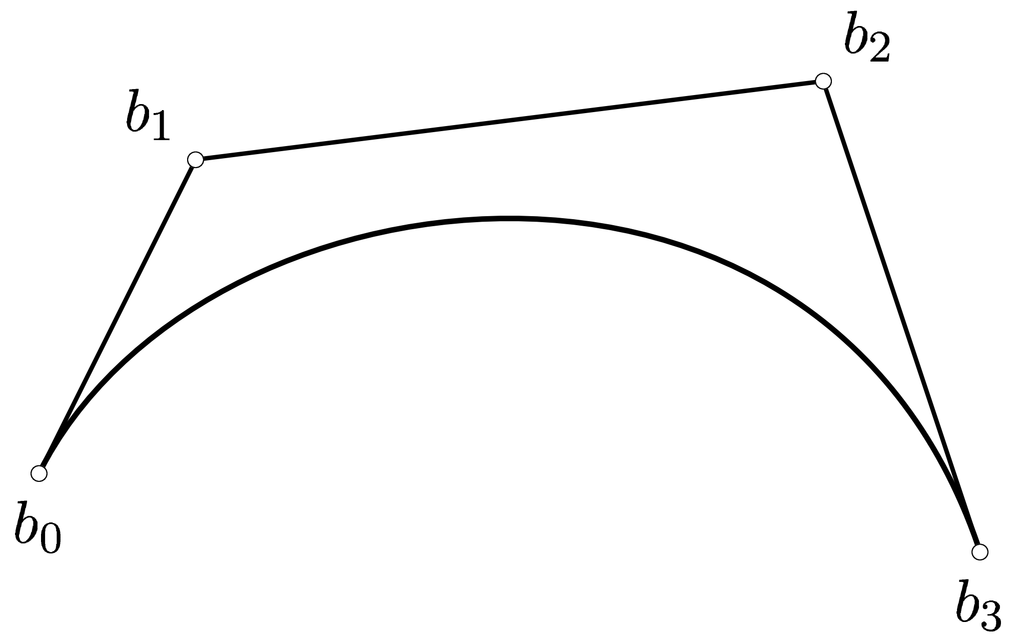

Figure 8: The (cubic) curve segment and its control polygon

Note that the curve goes through b_0 and b_3.

Constructing a point using affine interpolation

Surprisingly, there is a very simple and efficient method for constructing points on a curve specified by its control points.

This methods uses affine interpolation.

Given any two points a, b, and any real, $t \in \mathbb{R}$, recal that the (affine) interpolant

$$(1 - t)a + tb$$

is the point, c, on the line ab defined so that

$$c = a + t\overrightarrow{ab}.$$

This means that $c = (1 - t)a + tb$ is "t of the way between a and b".

Affine interpolation

For example, when $t = 1/2$, the point c is half way between a and b, i.e., it is the midpoint of ab.

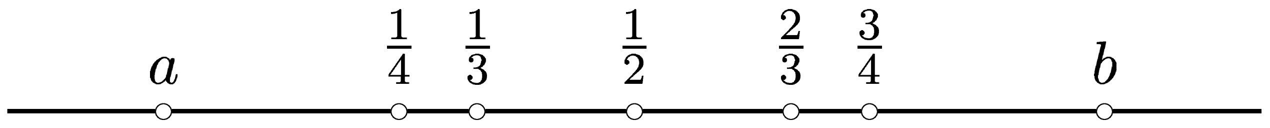

Figure 9: Some Affine interpolant for $t = 1/4, 1/3, 1/2, 2/3, 3/4$

If $t = 0$, then $c = a$ and if $t = 1$, then $c = b$.

Constructing a point using affine interpolation

When $t \notin [0, 1]$, the point $(1 - t)a + tb$ is outside of the line segment ab, we say talk about extrapolation rather than interpolation.

If a curve, F, is specified by control points, say b_0, b_1, b_2, b_3, for any $t \in [0, 1]$, we can construct the point $F(t)$ on F in three stages, using affine interpolation steps.

This process is called the de Casteljau algorithm.

5. The de Casteljau algorithm

Three stages. During stage 3, the last stage, we compute the point on the curve corresponding to t.

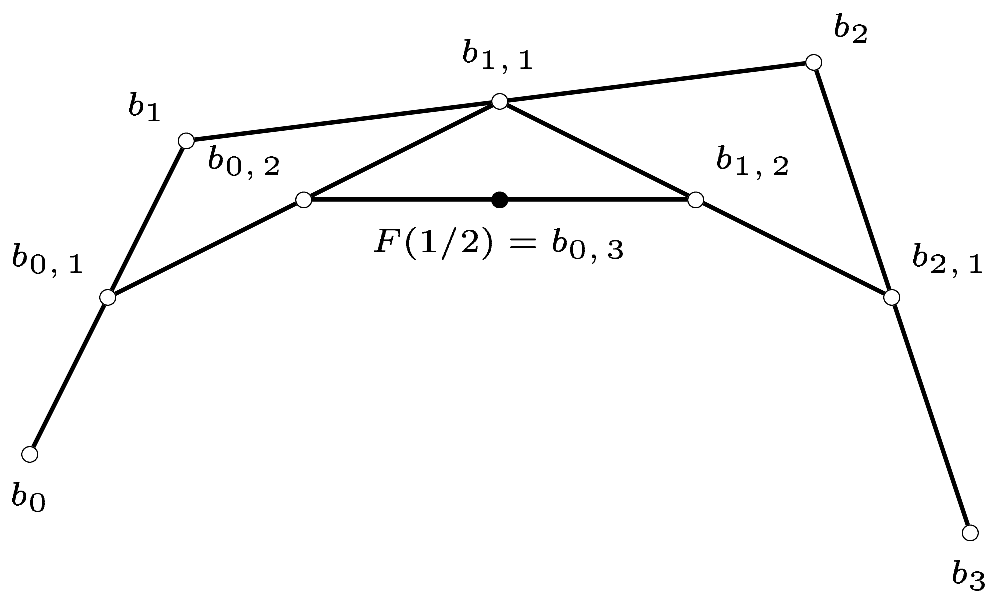

Figure 10: A de Casteljau diagram for $t = 1/2$

The subdivision method

If we look carefully at the method, say for $m = 3$, we discover that the process has constructed two control polygons

$$b_0, b_{0,1}, b_{0,2}, b_{0,3} \quad \text{and} \quad b_{0,3}, b_{1,2}, b_{2,1}, b_3.$$

So, we can subdivide recursively each of these control polygons.

This method yields a fast and efficient way of drawing a curve.

Indeed, it can be shown that these polygons quickly to the curve and that the "error" goes to zero at an exponential rate.

Example of The subdivision method

Here is an example of subdividing recursively six times:

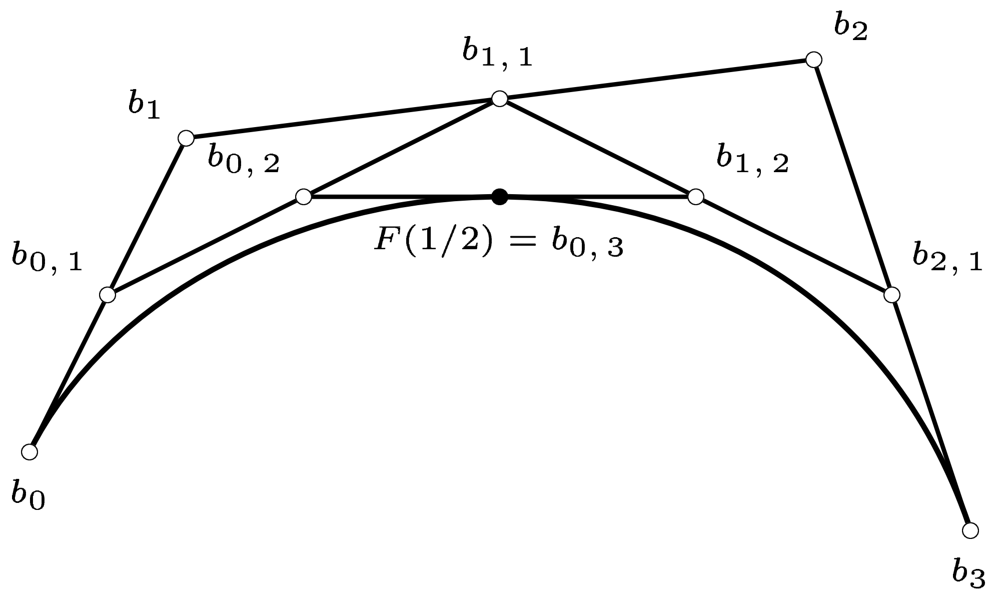

Figure 11: Approximating a curve using subdivision

Another example of subdivision

Here is another example of subdividing six times a curve of degree 4 given by the control polygon

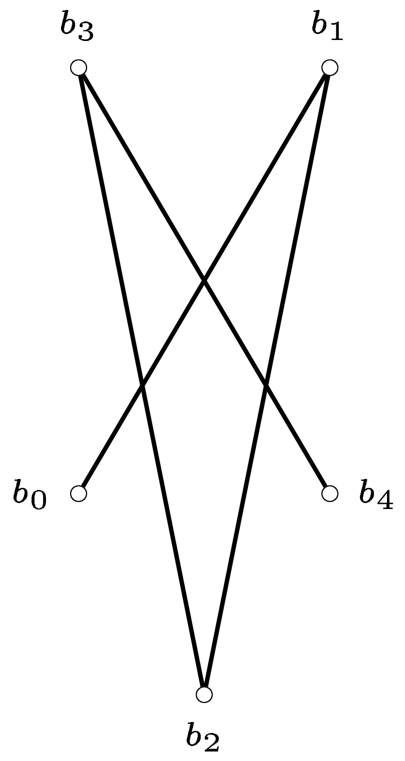

Figure 12: A control polygon for a curve of degree 4

Example of subdivision, one iteration

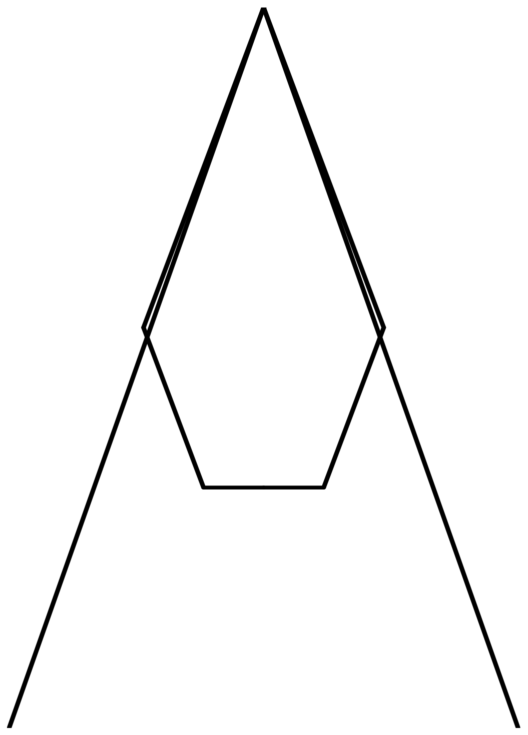

Figure 13: Subdivision, 1 iteration

Example of subdivision, two iterations

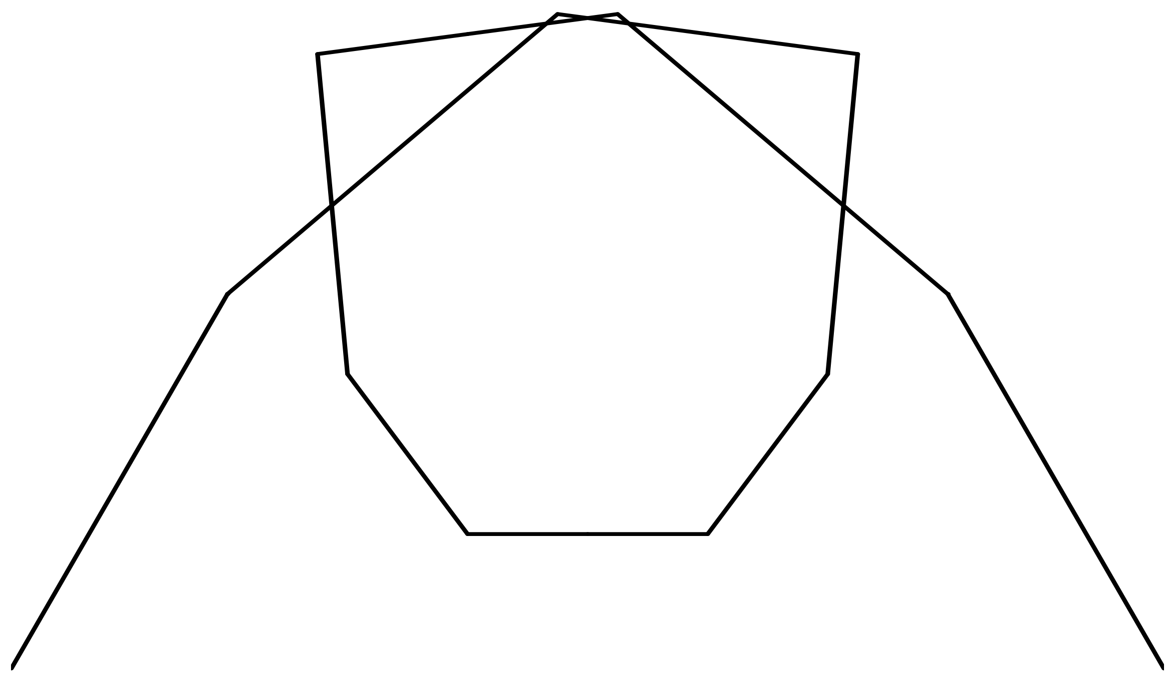

Figure 14: Subdivision, 2 iterations

Example of subdivision, three iterations

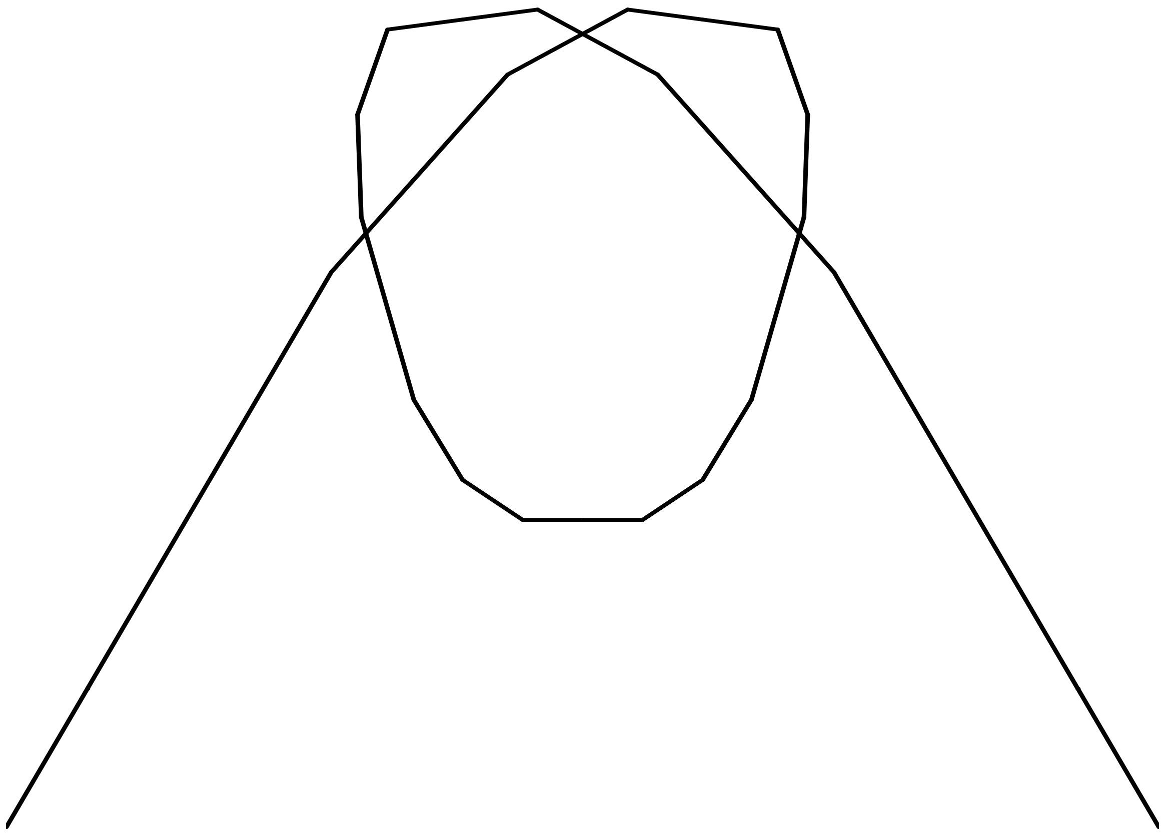

Figure 15: Subdivision, 3 iterations

Example of subdivision, four iterations

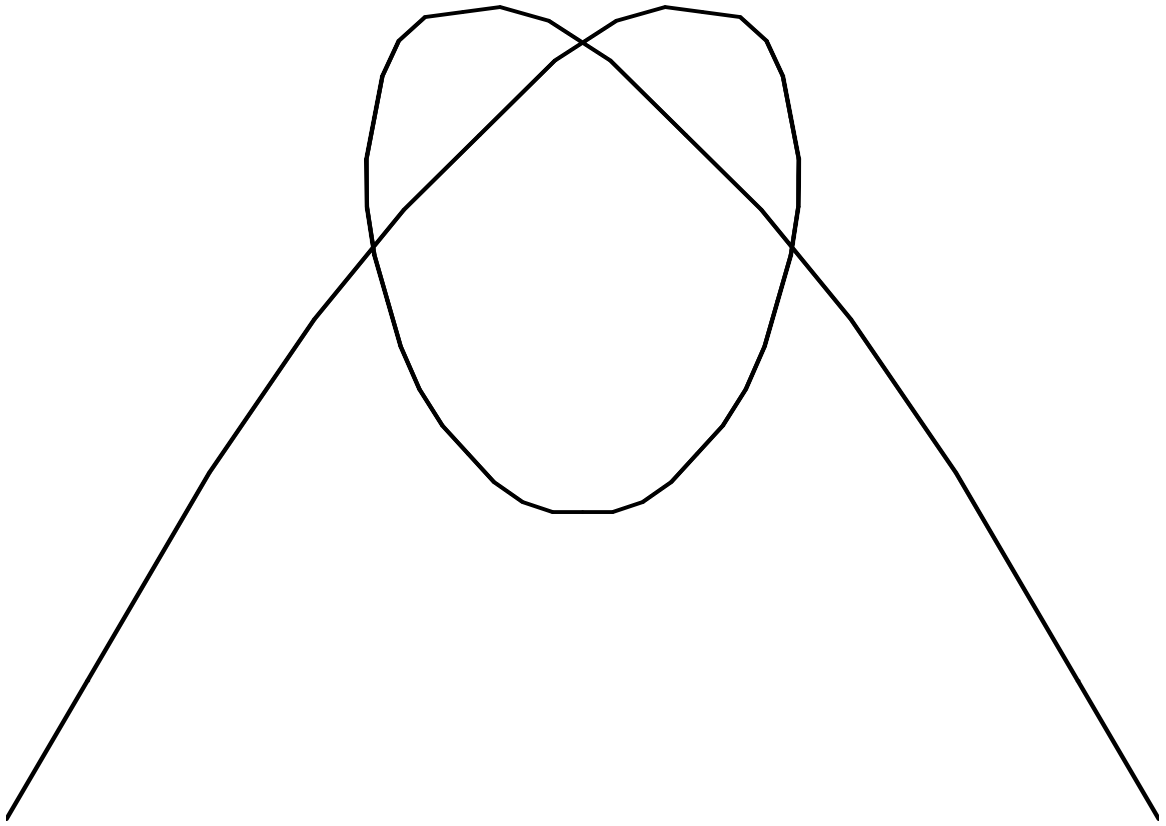

Figure 16: Subdivision, 4 iterations

Example of subdivision, five iterations

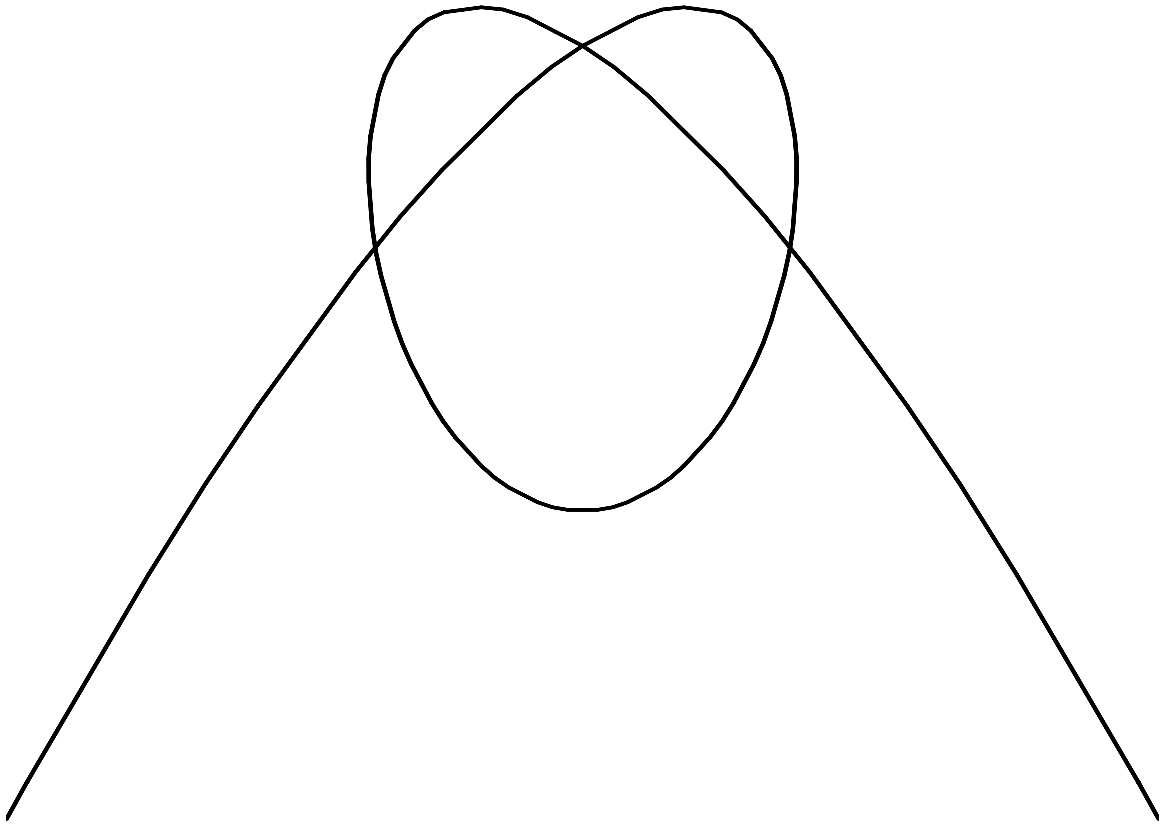

Figure 17: Subdivision, 5 iterations

Example of subdivision, six iterations

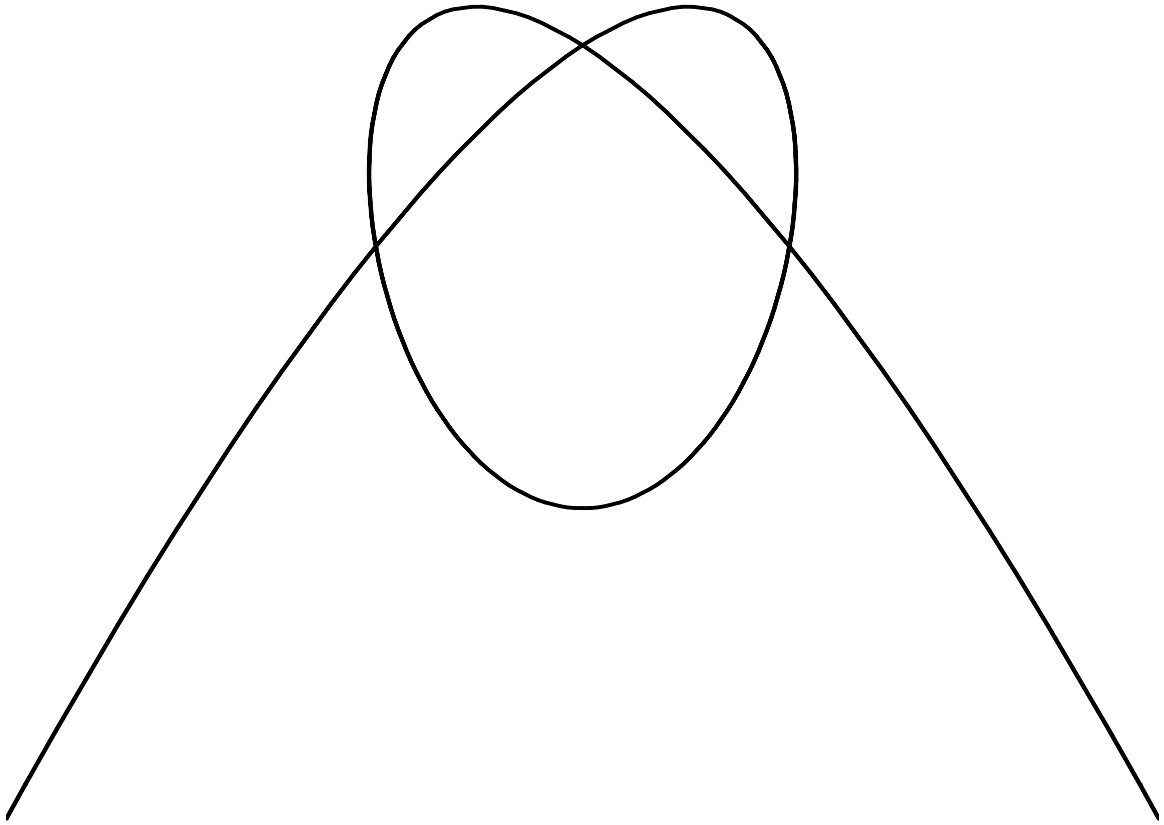

Figure 18: Subdivision, 6 iterations

6. Back to B-Spline Curves!

B-Splines are specified by a control polygon and a "knot sequence", specifying continuity between the curve segments.

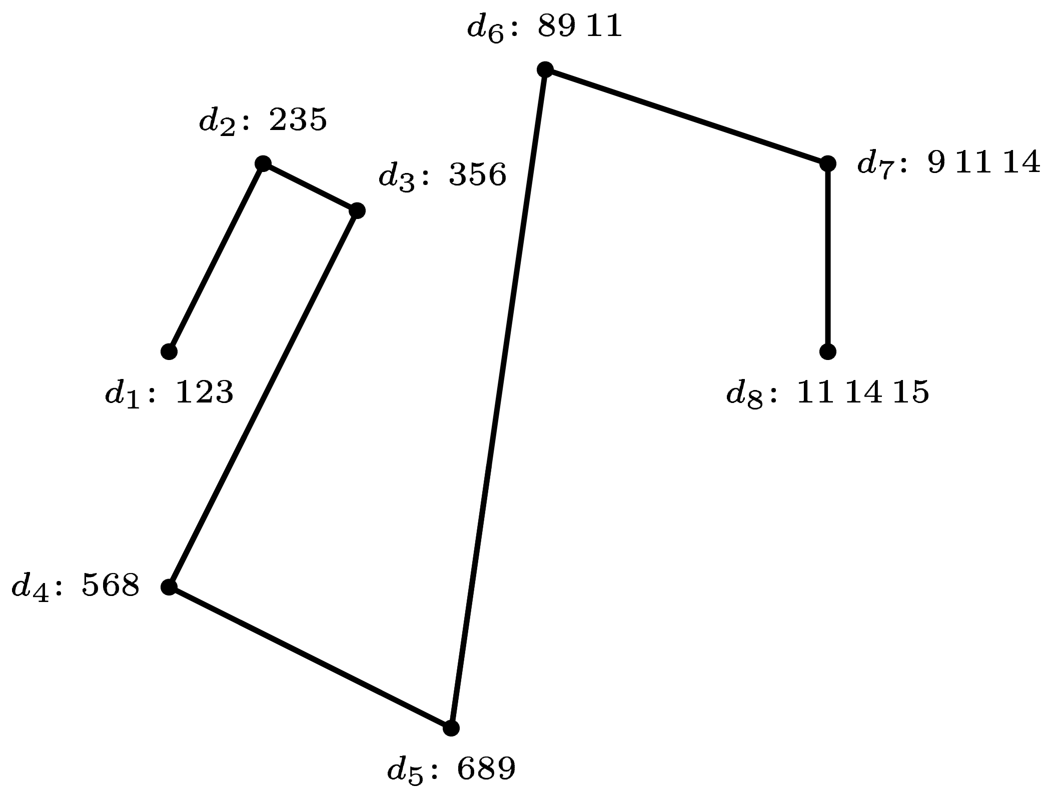

Figure 19: Control polygon of a cubic spline

Example of a Splines

The previous polygon defines the following spline with two endpoints:

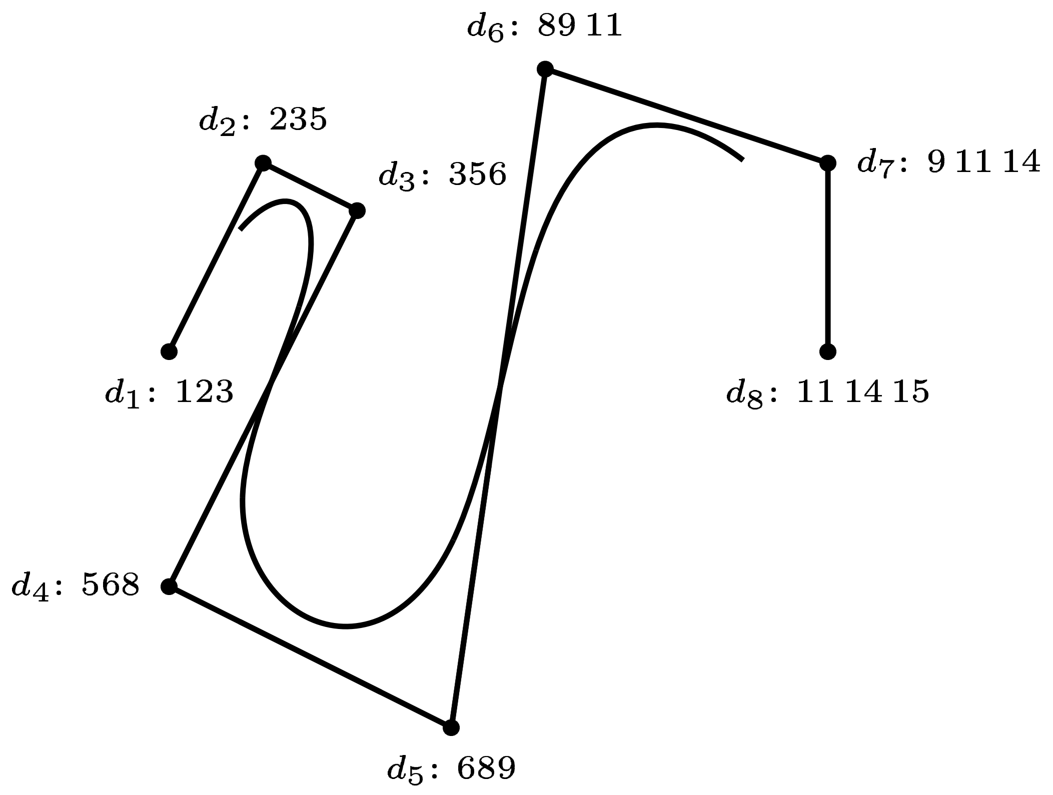

Figure 20: A cubic spline

Bézier Segments of a spline

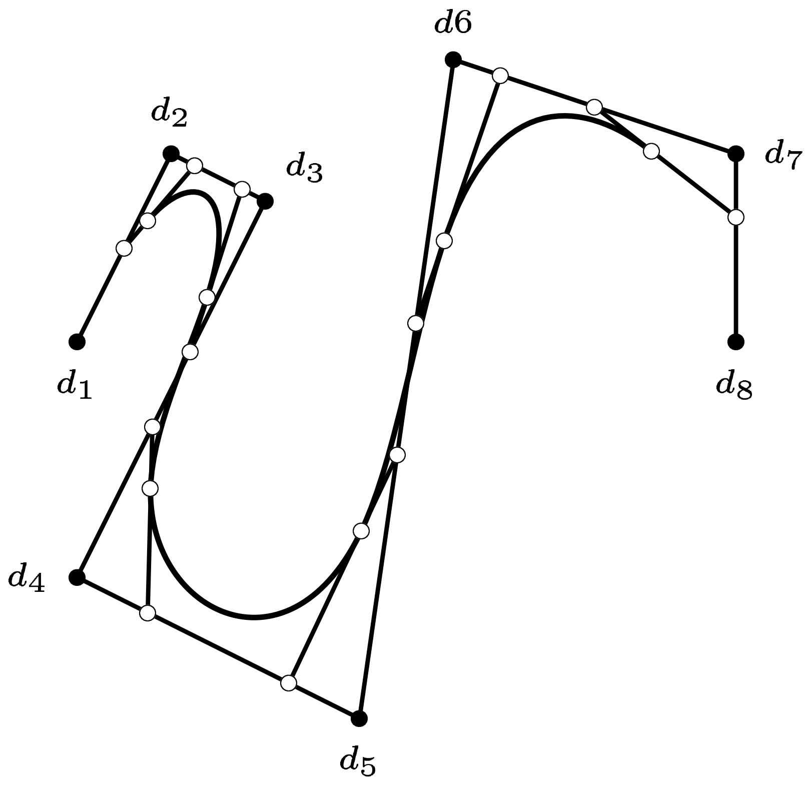

Figure 21: Bézier control points of a cubic spline

Closed Splines

Can we handle closed curves? No problem . Even for interpolation.

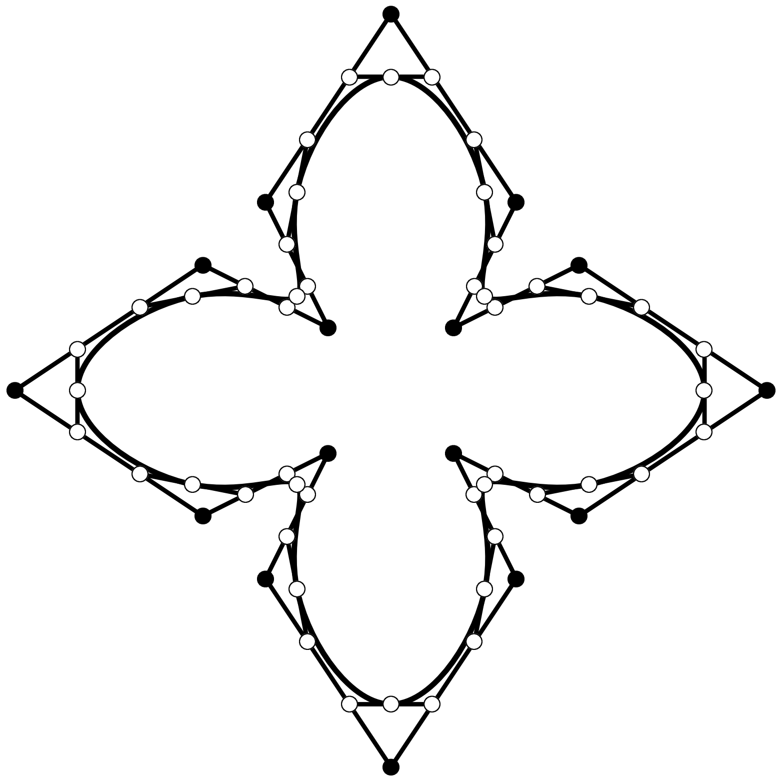

Figure 22: A closed cubic spline

7. Surfaces

A polynomial surface (in $\mathbb{R}^3$) is a function, F, of two variables, U, V, given by three polynomials:

$$\begin{aligned} x &= F_1(U,V) \\ y &= F_2(U,V) \\ z &= F_3(U,V). \end{aligned}$$

We can think of a surface as a curve of curves. If we hold U constant, then we obtain a curve (in the parameter V).

We can think of a surface as the result of deforming and stretching a flat infinite plane, without tearing it, but allowing creases and self-intersections.

Example of a Polynomial Surface

For example, the Monkey saddle is given by

$$\begin{aligned} x(U,V) &= U \\ y(U,V) &= V \\ z(U,V) &= U^3 - 3UV^2. \end{aligned}$$

Examples of a Surface

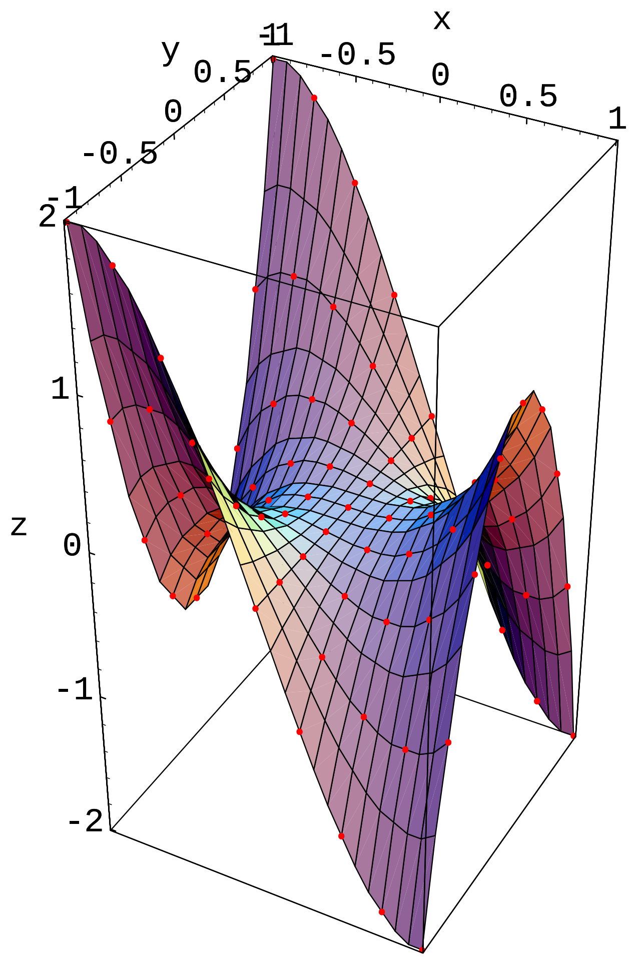

Figure 23: A monkey saddle

Rectangular and Triangular Patches

Usually, we are not interested in the whole surface, but only in a patch.

A patch is a piece of the surface whose domain is a rectangle or a triangle.

A rectangular patch is a patch whose domain is a rectangle.

A triangular patch is a patch whose domain is a triangle.

As in the case of curves, both rectangular and triangular patches can be defined in terms of control points. This time, we use control nets as opposed to control polygons.

Examples of a Rectangular Patch

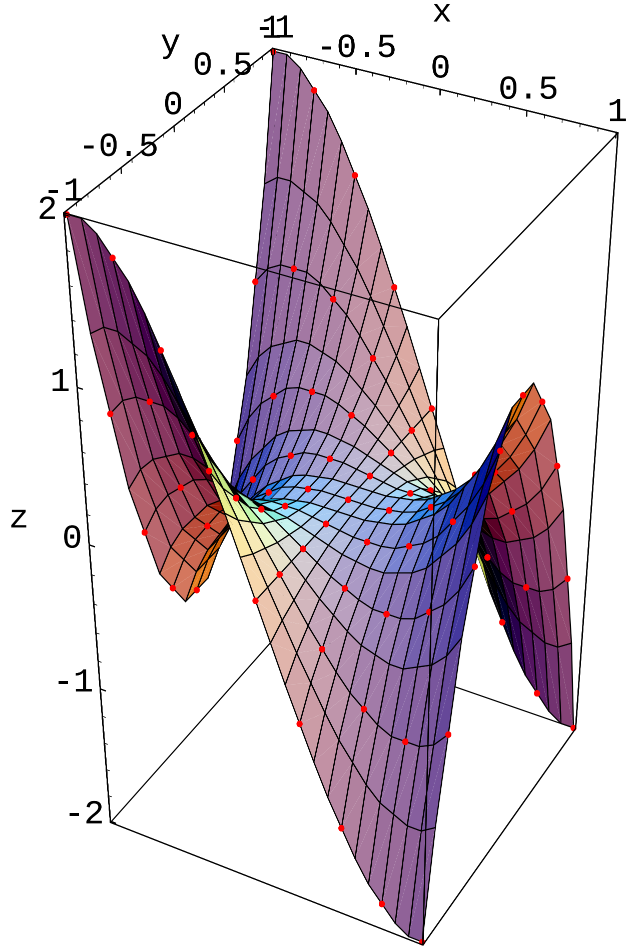

Figure 24: A monkey saddle

Example of a Triangular Patch

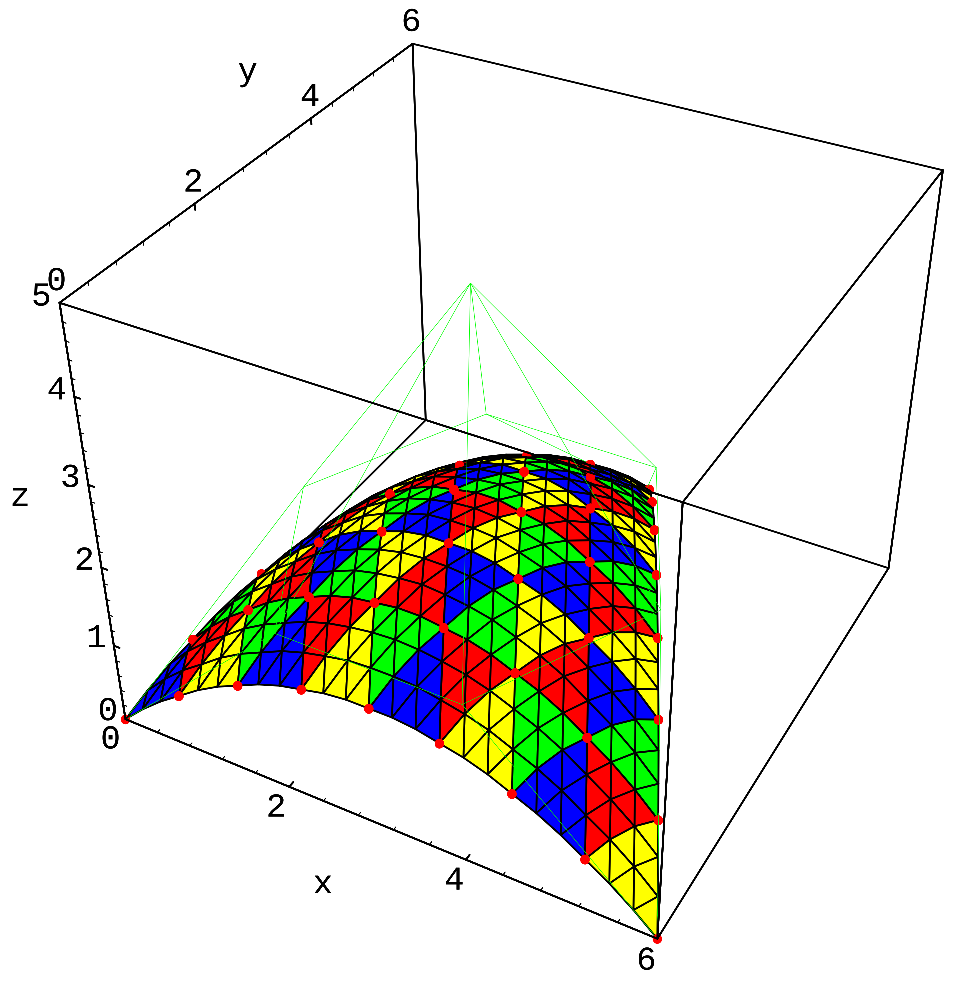

Figure 25: A triangular surface patch

De Casteljau Algorithm and Subdivision (Surfaces)

Both for rectangular and triangular patches, the de Casteljau algorithm works (constructing a point on the surface using affine interpolation).

It is an easy generalization of the curve case in the rectangular case, but it is a little more tricky in the triangular case.

The recursive subdivision method also works, but it is more tricky in the triangular case.

A triangle can be subdivided in four subtriangles by joining the midpoints of its edges.

Example of Subdivision (triangular patch)

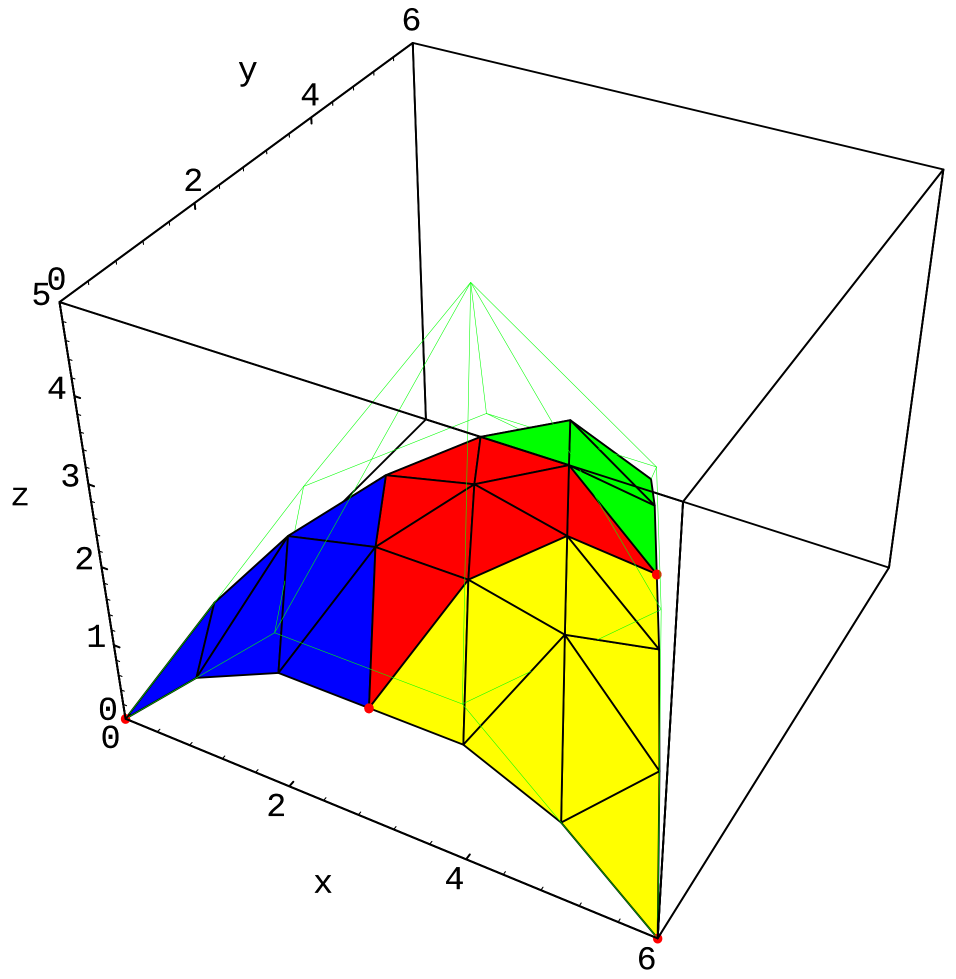

Figure 26: Subdivision, 1 iteration

Example of Subdivision (triangular patch)

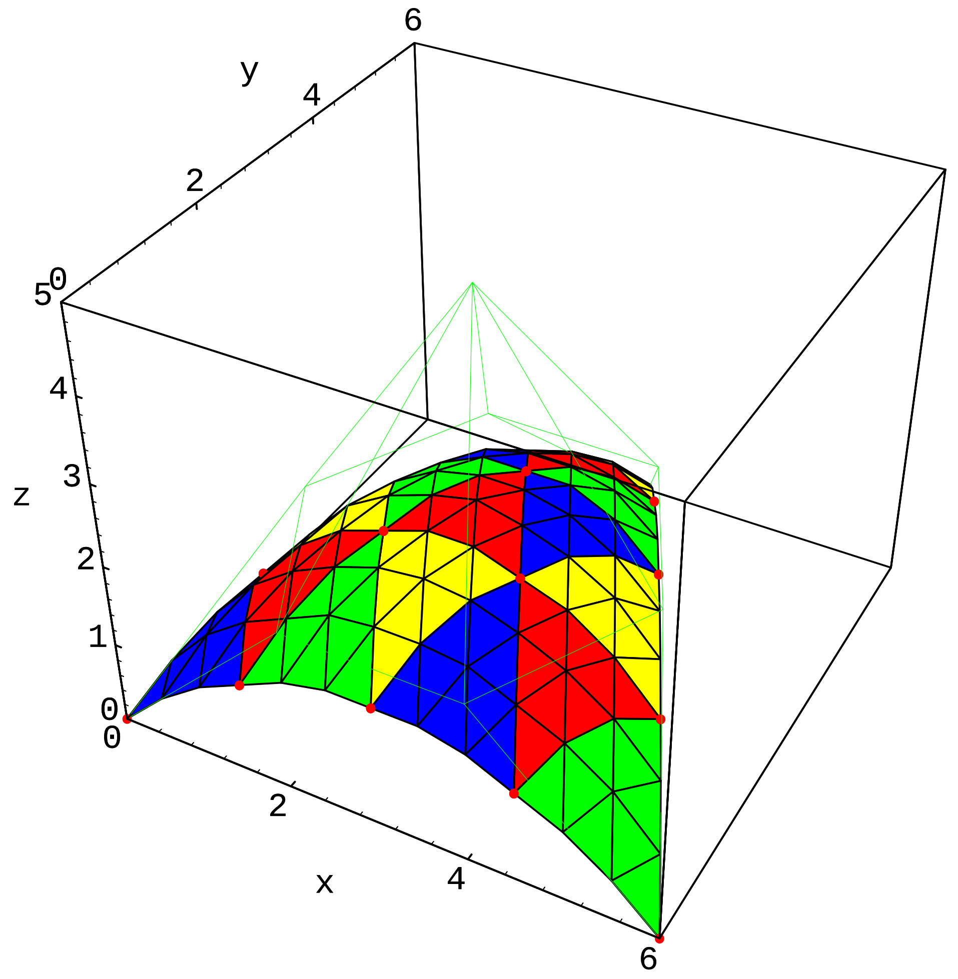

Figure 27: Subdivision, 2 iterations

Example of Subdivision (triangular patch)

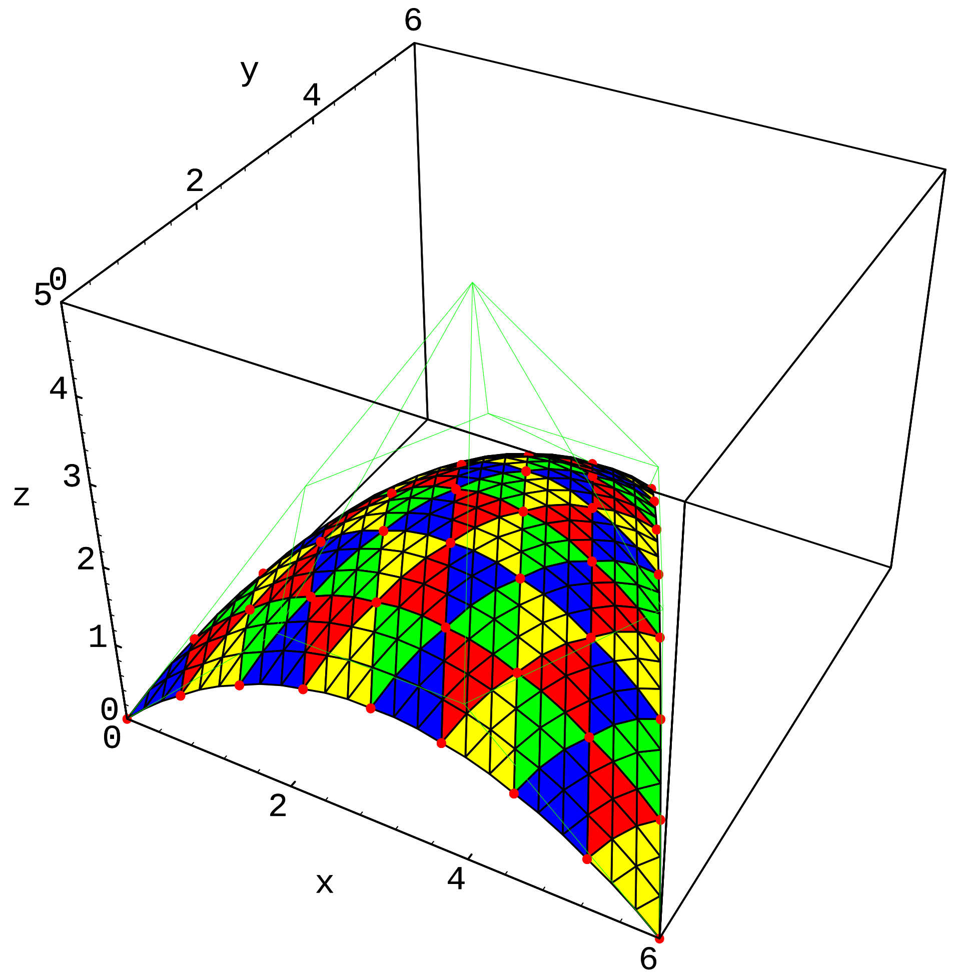

Figure 28: Subdivision, 3 iterations

Surface Splines

What about surface splines?

This depends on the way the domain (in the plane) is subdivided.

There are primarily two ways of subdividing the domain:

(1) Subdivide into rectangles

(2) Subdivide into triangles

Case (1) is easier because only four rectangles come together at a common vertex.

There is a well developed technology: NURBS!

Surface Splines

Case (2) (triangles) is a lot harder. Unfortunately, many triangles can share a common vertex.

Yet, rectangular splines are overly restrictive: Only certain kinds of surfaces can be represented and it is hard (or impossible) to deal with sharp corners or holes.

Triangular splines are more general and more flexible. Recent work by Dianna Xu and Gallier proposes new methods for dealing with triangular splines.

8. “Sculpting” and Subdivision Surfaces

An attractive alternative to surface splines is subdivision surfaces.

The basic idea is reminiscent of “sculpting”.

Start with a solid shape, called a mesh, and apply rules for refining and smoothing the mesh using a subdivision method.

New vertices, new edges and new faces are created by “smoothing corners”, as if we used a chisel.

The desired surface is a limit surface (see links in home page).

Subdivision Surfaces, Quick History

The idea of defining a curve or a surface via a limit process involving subdivision goes back to Chaikin (1974).

In 1978, two subdivision schemes for surfaces were proposed by Doo and Sabin and by Catmull and Clark.

Several years later, Charles Loop in his Master's thesis (1987) introduced a subdivision scheme based on a mesh consisting strictly of triangular faces.

Conclusion and Challenges

Subdivision surfaces are used in the animated movie industry (Pixar) but they are gaining popularity in other areas of geometic modeling.

See the links in the home page for the talk for more info and some demos on subdivision surfaces.

Their main drawback is that they are not parametric and it is hard to predict what the end-shape will be.

It would useful to marry the advantages of triangular splines (a la Xu/Gallier) and subdivision surfaces.

We're working on it and we need all the help we can get!

Some References

Curves and Surfaces for CAGD: A Practical Guide, by Gerald Farin, Academic Press, 2001, Fifth Edition.

Curves and Surfaces in Geometric Modeling: Theory and Algorithms, by Jean Gallier, Morgan Kaufmann, 1999, First Edition.

Geometric Modeling with Splines: An Introduction, by Elaine Cohen, Richard Riesenfed and Gershon Elber, A.K. Peters, 2001, First Edition.