Announcements

- Second assignment available on WWW.
- Replaced paper under *Animal Skin* topic.
- Thursday: fire and smoke. *HEPL*.
- Tuesday: finish rational curves, tensor product, NURBS surfaces. *CIS*.

The second assignment is a bit harder than the first one. We have already covered all you need to do it, so start now. Doing the assignment ASAP will also help you follow the lectures more closely.

The new paper is being handed out today. Leftover copies will be left on my desk. The figures of the new paper were selected for the cover of the SIGGRAPH '95 proceedings — it would be a pity not to cover it. The replaced paper by Kajiya is superseded by Perlin's work, which is part of the *Animal Skin* presentation; nevertheless, Kajiya's paper is still useful for those who lack sufficient background on volume rendering.

Note the different meeting place for this Thursday. Are Andrea Rosso, Rob Engle, and Jeffrey Chan here?

Most images from the papers, as well as some animations, are on the WWW. I also brought the papers to class to see the pictures.

Today

- B-spline curve topics:
  - Closed curves.
  - Shape refinement.
- Rational curves:
  - Definition and specification.
  - Drawing algorithm.
  - Degree 2 rational curves.
  - Weight redundancy.

Closed Splines

First and last segment meet: first segment and segment after last overlap. Curve becomes periodic, i.e.

$$\mathbf{p}(t) = \mathbf{p}(t+T)$$

where T is difference of first and last knot.

Continuity achieved by sharing de Boor points between first and last segment.

Drawing steps:

1. Extend knot sequence.

2. Label de Boor points circularly.

3. Assign de Boor points to segments.

4. Draw segments.

We will not go into much detail, here. The goal of our discussion on this topic is to provide some fundamental intuition, not a detailed algorithm in excruciating detail. To this end, we'll show a single example illustrating the drawing steps listed on the slide.

The last two steps are performed using the familiar algorithms for point-to-segment assignment and de Boor interpolations; they are only listed here for completeness.

Closed Splines: Knot Sequence

Step 1: Given knots 0, 1, 2, 3 with multiplicities 1, 1, 1, form doubled knot sequence.

1. Write usual sequence excluding last knot:

$$0, 1, 2.$$

2. Find difference T of first and last knots:

$$T = 3 - 0 = 3.$$

3. Shift sequence from step 1 by T and append result to original sequence:

$$0, 1, 2, \ \mathit{3}, \ \mathit{4}, \ \mathit{5}.$$

The given closed spline will have three segments: $0 \ldots 1$, $1 \ldots 2$, and $2 \ldots 3$. The last segment and the first will be adjacent, and hence, the segment following the last one, namely $3 \ldots 4$, will overlap the first one, $0 \ldots 1$. The intuition behind extending the knot sequence is that, in drawing the spline, we'd like to draw the first segment by using 3 and 4 as its endpoints, rather than using 0 and 1 — the extended sequence will provide the necessary labels for the de Boor points of segment $3 \ldots 4$.

Since the curve is closed, knots 0 and 3 correspond to the same joint, where segments $2 \ldots 3$ and $0 \ldots 1$ (a.k.a. $3 \ldots 4$) meet; alternatively, the periodicity requirement implies $\mathbf{p}(0) = \mathbf{p}(3)$. So, when the user sets the multiplicity of the first knot, the continuity between the first and last segment is defined; here, $2 \ldots 3$ and $0 \ldots 1$ meet with C^1 continuity.

The method by which we proceed to replicate the knot sequence guarantees that we use the same multiplicity for knots 0 and 3 — this makes sense given that both knots correspond to the same joint. Also, our replication identifies the automatically created knots 4 and 5 with the given knots 1 and 2, respectively — as in the case of knots 0 and 3, knots 4 and 1 correspond to the same joint, and similarly for knots 5 and 2.

Closed Splines: Labels

Step 2: Given three de Boor points A, B, C, assign labels circularly.

Point	Labels
A	01, 34
B	12, 45
C	23

We first assign one row to each de Boor point, in order. Then, we labeled them by going down from top to bottom and starting again on the first row once we have passed the last one.

This circular assignment of labels has the effect that some de Boor points have duplicate labels. This implies that, when we assign de Boor points to segments, segments $3 \ldots 4$ (drawn in place of $0 \ldots 1$) and $1 \ldots 2$, will end up sharing as many de Boor points as their desired continuity demands...

Closed Splines: Assignments

Step 3: Assign de Boor points to segments as usual.

Segment	0	1	2	3	4	5
	0	1				
$1 \ldots 2$		1	2			
			2	3		
		1	2			
$2 \ldots 3$			2	3		
				3	4	
			2	3		
$3 \ldots 4$				3	4	
					4	5

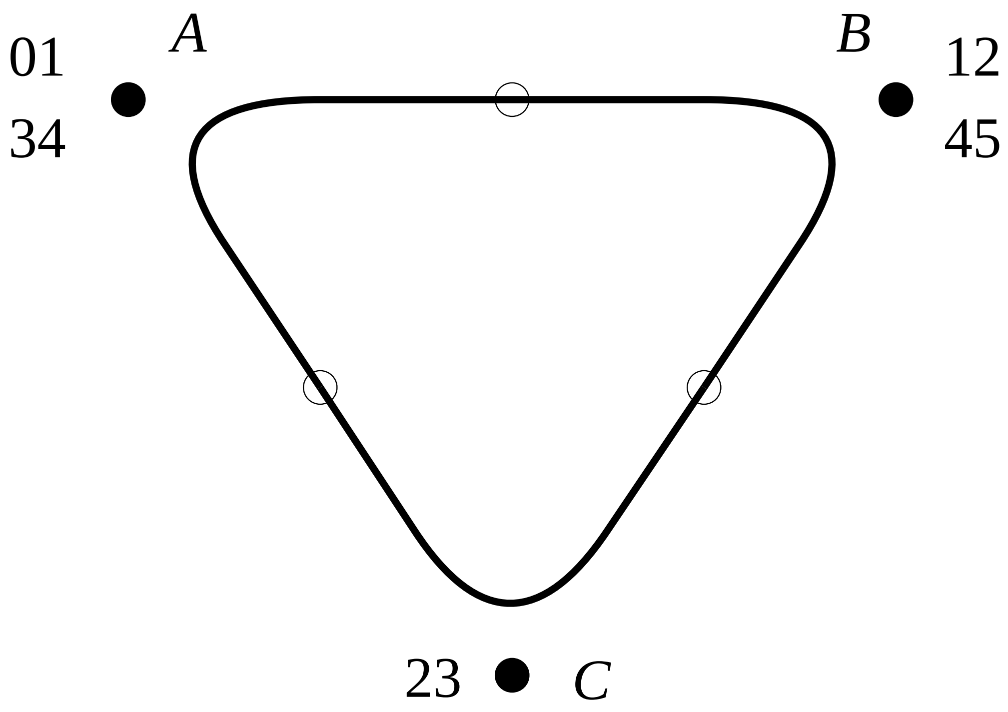
In this table, we took advantage of the fact that segments $0 \ldots 1$ and $3 \ldots 4$ overlap. In particular, we did not list segment $0 \ldots 1$ since we would not be able to assign it three de Boor points, given our labeling scheme; instead, our knot sequence extension and labeling scheme ensure that the first segment will be drawn as $3 \ldots 4$.

Observe how our labeling scheme ensured that segments $1 \ldots 2$ and $3 \ldots 4$ share two de Boor points, as C^1 continuity demands: both labels 01 and 34 belong to de Boor point A and both labels 12 and 45 belong to B.

In order for this assignment of de Boor points to segments to be feasible, we *had* to extend the original knot sequence, otherwise we would have run out of points for the first and last segment of the spline ($0 \ldots 1$, a.k.a. $3 \ldots 4$, and $2 \ldots 3$). In this example, we duplicated the knot sequence, but, in general, we need only shift and repeat a few of its initial elements.

Closed Splines: Drawing

Step 4: Draw segments as usual.



We drew the spline using the familiar de Boor algorithm and the assignment of de Boor points to segments shown on the previous slide. The solid disks represent the de Boor points, and the open circles represent the (derived) Bézier points.

As a minor remark, observe that the statement "1 ... 2 and 3 ... 4 meet with C^1 continuity" is, strictly speaking, incorrect. Instead, it is 3 ... 4 and 4 ... 5 which meet with C^1 continuity at $t = 4$. As it should be obvious by now, the reason we use the incorrect informal statement is that segments 4 ... 5 and 1 ... 2 appear identical — that is, they are reparameterizations of the same curve. However, the strict definition of C^1 continuity requires agreement of position (C^0 continuity), and at $t = 4$, segment 1 ... 2 does not yield the point halfway between de Boor points A and B — the open circle, which is what 3 ... 4 yields at $t = 4$.

Shape Refinement

Problem: Given B-spline curve, find new representation which

- initially describes given shape, and
- provides more flexibility in changing shape.

Solutions:

Degree raising: increase degree of spline by one.

Knot insertion: insert a knot (or increase multiplicity of existing knot).

Both solutions increase number of de Boor points; hence, more freedom in design.

Shape refinement is very useful. For example, when the designer draws his/her first draft of a car's outline, s/he wants to get a rough sketch, and thus uses few de Boor points, and (at most) a cubic spline. The reason for using few de Boor points is that, for a rough model, extra de Boor points are useless and get in the way; why should the user have to specify more parameters when fewer can get the job done? Higher degree splines are undesirable because, for the same number of segments, a higher degree spline requires more de Boor points than a lower degree spline; also, higher degree splines introduce more overhead (we will discuss this later). Shape refinement becomes necessary once the design is approved by marketing, and the engineer must refine the curve to optimize aerodynamics: such refinement requires additional, finer control. How can we increase our control over a given spline?

"Degree raising" is also known as *degree elevation*.

Degree Raising

1. Increase all knot multiplicities by one for adjacent segments to maintain their continuity.

   Why? Example:

   - Affine spline with C^0 joint has knot multiplicity 1.

   - Quadratic spline with C^0 joint has knot multiplicity 2.

2. Raise the degree of each segment separately.

   How? Similar algorithm to degree-raising Bézier curves.

As before, the goal of our discussion on this topic is to provide fundamental intuition, not a full recipe.

Recall that, in a degree d spline, a knot multiplicity of n guarantees C^{d-n} continuity.

Degree Raising: Knot Sequence

Given:

Segment	00	11	2	33
$0 \ldots 1$	00 0	 1 11		
$1 \ldots 2$		11 1	 2 2	 3
$2 \ldots 3$		1	2 2	 3 33

This is a quadratic spline — that's why each de Boor point is labeled with two adjacent elements in the knot sequence. We will turn it into a cubic spline. First, we'll generate the new knot sequence and assign de Boor points to the new (cubic) segments. Then, we'll figure out the positions of the new de Boor points.

Degree Raising: Knot Sequence

Result:

Segment	000	111	22	333
$0 \ldots 1$	000 00 0	 1 11 111		
$1 \ldots 2$		111 11 1	 2 22 22	 3
$2 \ldots 3$		1	22 22 2	 3 33 333

We have increased the knot multiplicities and assigned de Boor points to segments in such a way that the same continuity is maintained between segments: C^0 between $0\ldots1$ and $1\ldots2$ and C^1 between $1\ldots2$ and $2\ldots3$.

Degree Raising: de Boor Points

Degree-raise each segment separately, e.g.

- $1 \ldots 2$:

$$\hat{\mathbf{p}}_1(1,2,2) = \frac{1}{3}[\hat{\mathbf{p}}_1(2,2) + 2\hat{\mathbf{p}}_1(1,2)].$$

  Find $\hat{\mathbf{p}}_1(2,2)$ from $\hat{\mathbf{p}}_1(1,2)$ and $\hat{\mathbf{p}}_1(2,3)$.

- $2 \ldots 3$:

$$\hat{\mathbf{p}}_2(1,2,2) = \frac{1}{3}[\hat{\mathbf{p}}_2(2,2) + 2\hat{\mathbf{p}}_2(1,2)].$$

  Find $\hat{\mathbf{p}}_2(2,2)$ from $\hat{\mathbf{p}}_2(1,2)$ and $\hat{\mathbf{p}}_2(2,3)$.

Consistent algorithm:

$$\hat{\mathbf{p}}_1(1,2,2) = \hat{\mathbf{p}}_2(1,2,2)$$

because

$$\hat{\mathbf{p}}_1(1,2) = \hat{\mathbf{p}}_2(1,2) \quad \text{and} \quad \hat{\mathbf{p}}_1(2,3) = \hat{\mathbf{p}}_2(2,3).$$

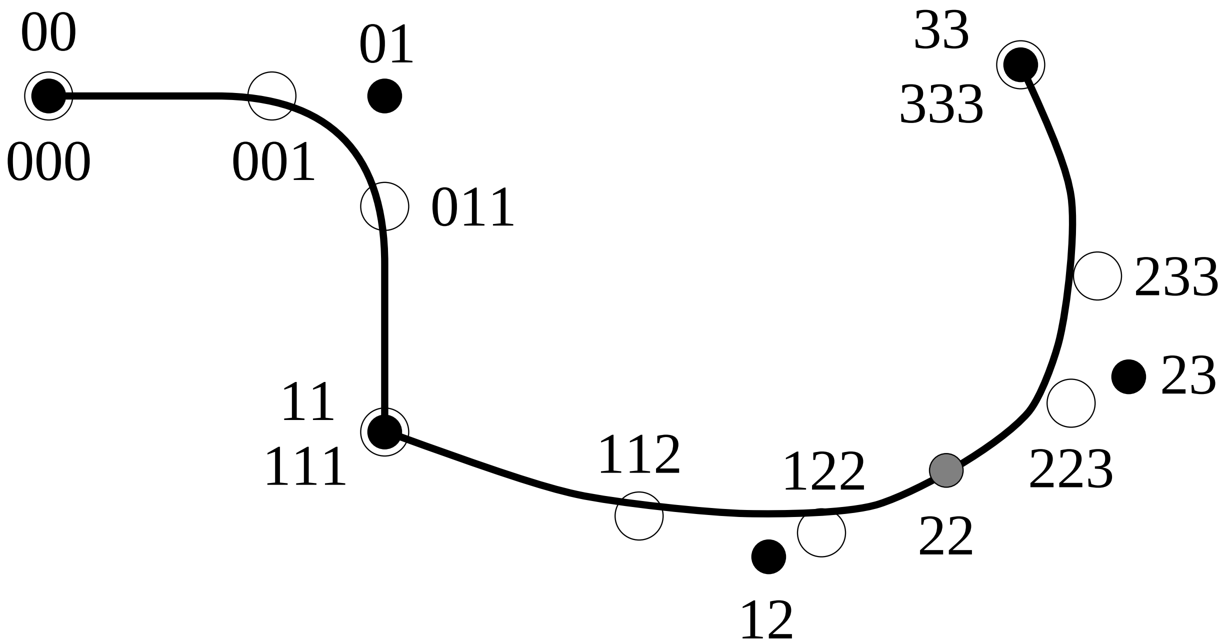
Recall that a quadratic Bézier curve $\mathbf{p}(t)$ is turned into a cubic one by using polar forms:

$$\widehat{\mathbf{p}}(t_1, t_2, t_3) = \frac{1}{3}[\widehat{\mathbf{p}}(t_1, t_2) + \widehat{\mathbf{p}}(t_1, t_3) + \widehat{\mathbf{p}}(t_2, t_3)].$$

When we raise the degree of spline segments, some of the terms on the right-hand side might be unavailable. However, all missing terms can be found by applying the general de Boor algorithm (yielding polar values for arbitrary polar arguments) to the original (quadratic) de Boor points of the segment.

Consistency is important. In the above example, the spline can only have a single de Boor point labeled 122. However, two (or more, in general) segments generate that new de Boor point by degree raising. Do they all get the same answer; that is, do they all agree on their polar values for arguments 1, 2, and 2? Yes, because all those segments which need the new point (here, 122) must have shared the original de Boor points (those points which were interpolated to produce the new one, here 12 and 23).

Degree Raising: de Boor Points



The solid disks represent the original de Boor points, and the open circles represent the new ones. It's easier to see how the new de Boor points labeled 122 and 223 were derived by first drawing the original Bézier point labeled 22: it falls halfway between the original de Boor points 12 and 23, and is drawn as a gray disk.

Degree Raising: Drawback

Must raise degree of *all* segments, even if we only want *local* refinement. Consequences:

- Inefficient use of memory.
- Slower drawing:
  - more interpolations, and
  - cache fills up faster.
- Slower analysis (curvature, derivatives, etc.).

The reason we must raise the degree of all the segments is, simply, that all the segments of a spline must have the same degree. There is a subtlety here: raising the degree of a spline as we did does not change its shape. Therefore, even though we now call a degree-raised quadratic a "cubic", it's nevertheless a degenerate cubic and thus still a quadratic in essence. The issue of concern here is that the computer now stores and processes a cubic representation, without being aware of the degeneracies. Thus, after some local modifications are made to the cubic spline, the degenerate cubics which were not modified will still be stored, drawn, and analyzed as cubics, not as quadratics.

Let us elaborate more on our observations:

- Higher degree splines require more de Boor points to be represented, and also longer knot sequences (because we must modify the knot multiplicities to preserve the original continuity).

- Drawing takes longer because the de Casteljau algorithm must perform more interpolations. Also, cache performance may plunge due to the larger working set (consisting of the spline's de Boor points and its knot sequence).

- Similarly, other curve analysis algorithms also take longer for higher-degree curves.

Knot Insertion

Insert a knot, i.e. split a segment into two segments.

1. Modify knot sequence.

   How? Insert new knot.

2. Find new de Boor points from original segment.

   How? Use simple interpolations.

Exactly same algorithm for increasing a knot's multiplicity by one.

Knot insertion allows us to perform local shape refinement without much extra cost. Since the degree of the spline stays the same, the addition of a knot (or an increase in a knot's multiplicity) just increases the number of the spline's de Boor points by one. This extra de Boor point appears in the vicinity where the curve needs to be tweaked, allowing for the desired local shape refinement.

The wording above may cause some confusion: the effect of knot insertion on the spline's de Boor points is not just the addition of a de Boor point. In particular, some of the original de Boor points (as many as the degree of the spline reduced by one) also have to be moved around. Why? Because changing the knot sequence changes the de Boor point labels; hence, in order for spline segments to retain their shape, we have to specify new polar values (the new positions of the de Boor points) for the new polar arguments (the de Boor point labels).

Knot Insertion: Knot Sequence

Given:

Segment	...	0	2	4	6	7	8	...
		0	2	4				
$4 \ldots 6$			2	4	6			
				4	6	7		
					6	7	8	

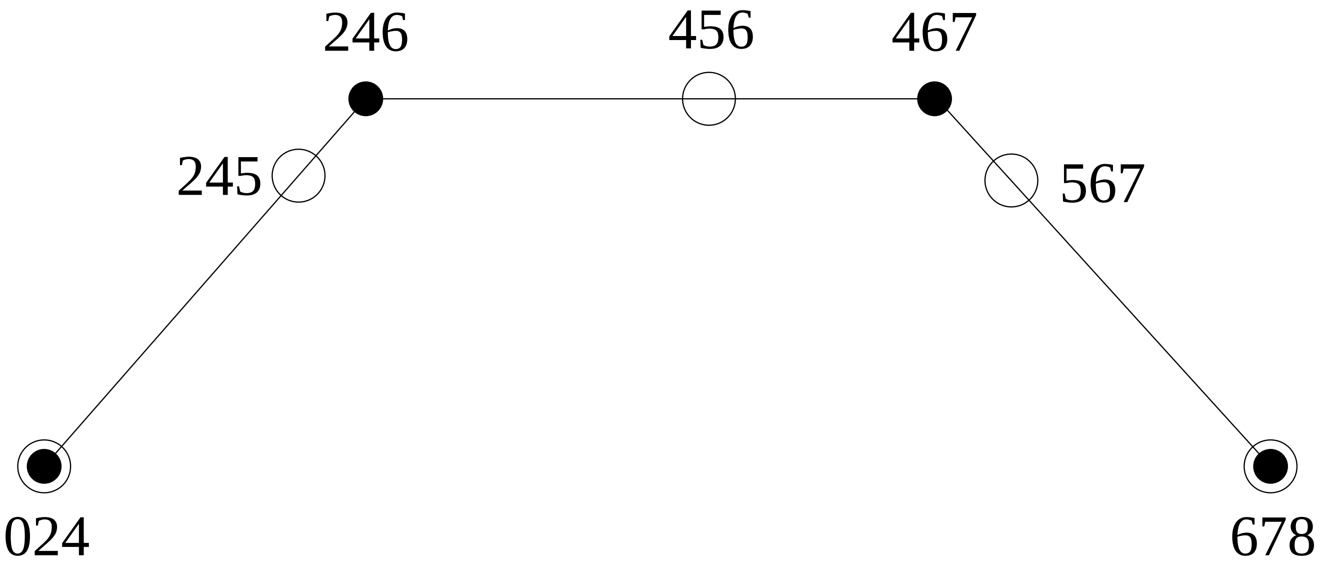
Result:

Segment	...	0	2	4	5	6	7	8	...
		0	2	4					
$4 \ldots 5$			2	4	5				
				4	5	6			
					5	6	7		
			2	4	5				
$5 \ldots 6$				4	5	6			
					5	6	7		
						6	7	8	

We are inserting a knot whose value is 5. The effect is splitting segment $4 \ldots 6$ into two segments $4 \ldots 5$ and $5 \ldots 6$.

Although not shown, there are other segments, besides the one split, which are now assigned different de Boor points. For example, segment $2 \ldots 4$ contained the de Boor point labeled 246; after inserting knot 5, it will contain the one labeled 245.

Knot Insertion: de Boor Points



$$\left.\begin{matrix}024\\246\end{matrix}\right\}245 \qquad \left.\begin{matrix}246\\467\end{matrix}\right\}456 \qquad \left.\begin{matrix}467\\678\end{matrix}\right\}567$$

The solid disks represent the original de Boor points, and the open circles represent the new ones.

The bottom line shows which two original de Boor points we used to find each of the new ones. Overall, we substituted two original de Boor points by three new ones, for a net addition of one de Boor point: this extra point gives us the additional flexibility needed to locally tweak the spline.

Finding the new de Boor points this way guarantees that the shape of the spline won't change. For example, segment $2\ldots4$, whose last de Boor point was labeled 246, now contains 245. But we got 245 from 246 and 024, and 024 did and does belong to $2\ldots4$. In other words, segment $2\ldots4$ was (partly) defined in terms of its polar values for polar arguments 0, 2, 4 and 2, 4, 6; now it's defined in terms of 0, 2, 4 and 2, 4, 5, where the last polar value was derived from the previous ones. It follows that the polar form of the segment $2\ldots4$ was not modified, and thus neither the shape of the curve. Similar argumentation for the other segments allows us to conclude that the overall shape of the spline was not modified.

Rational Curves

Parametric polynomials cannot represent sections of circles: as $t \to \infty$, all coordinates go to $\pm\infty$ (or always stay constant).

So, B-splines, open or closed, can only represent parabolas of all conic sections — *no ellipses, circles, hyperbolas*.

Definitions: Circle

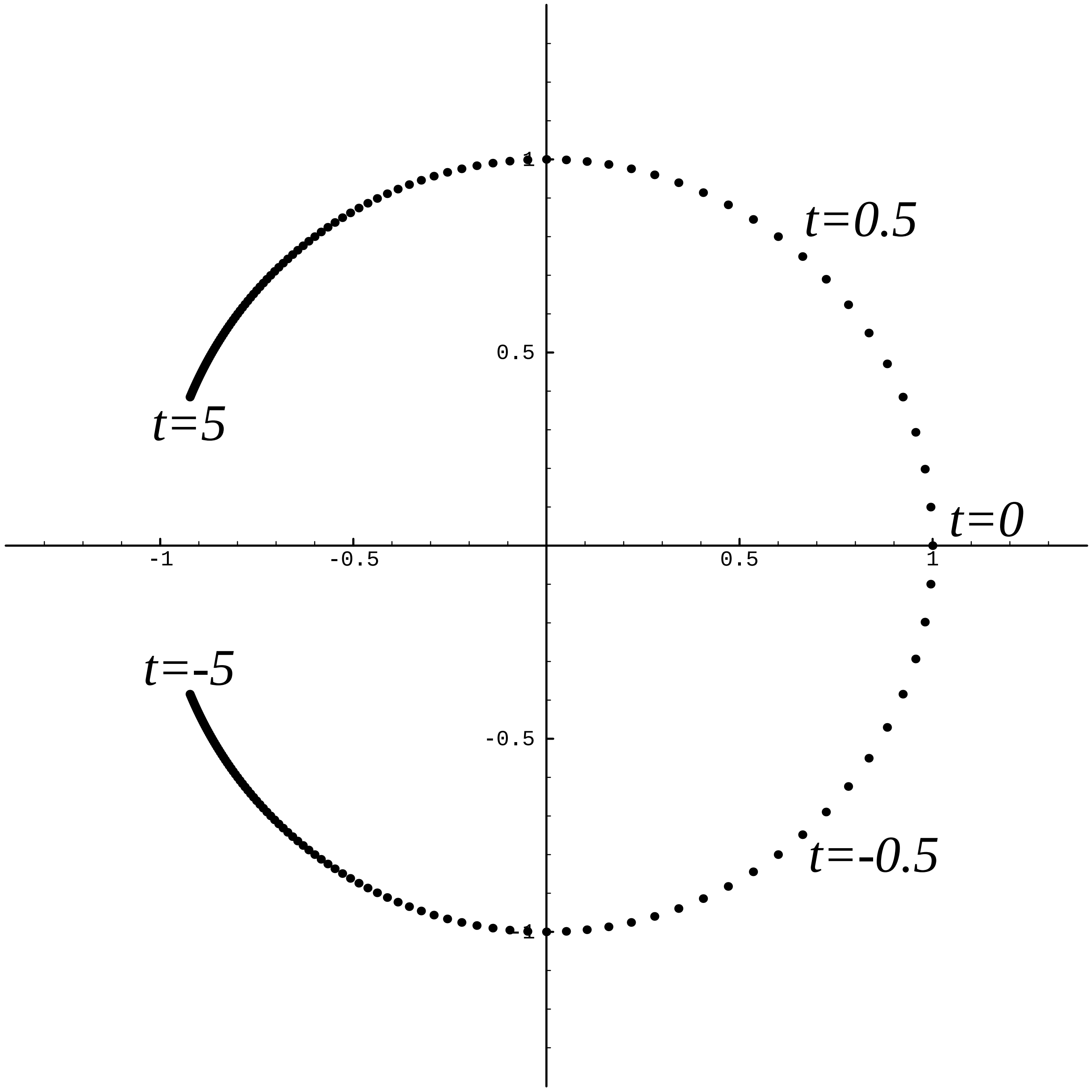
Circle can be represented by *ratios* of polynomials:

$$\mathbf{p}(t) = (p_x, p_y) = (\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}).$$

Verify:

$$\begin{aligned} & p_x^2 + p_y^2 \\ & \quad = \left(\frac{1-t^2}{1+t^2}\right)^2 + \left(\frac{2t}{1+t^2}\right)^2 \\ & \quad = \frac{1}{(1+t^2)^2}(1 + t^4 - 2t^2 + 4t^2) \\ & \quad = \frac{1}{(1+t^2)^2}(1 + t^4 + 2t^2) \\ & \quad = 1. \end{aligned}$$

Definitions: Circle



Note that the parameterization is highly non-uniform. We'll address this shortly.

Definitions: Circle

Can perceive circle

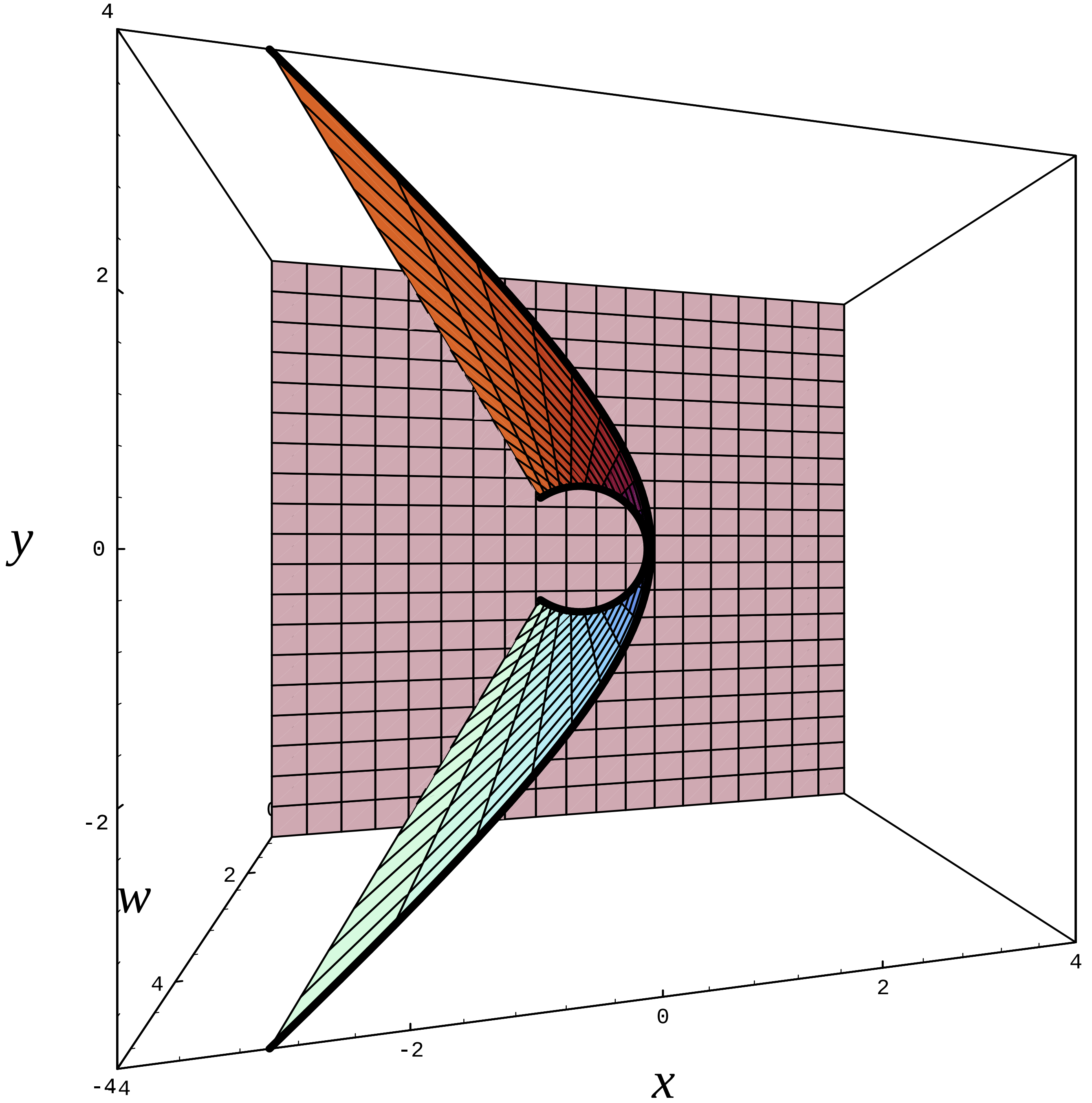
$$\mathbf{p}(t) = (\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2})$$

as curve in 3D space (w, x, y)

$$\mathbf{q}(t) = (1+t^2, 1-t^2, 2t)$$

projected on plane $w = 1$ through origin.

Definitions: Circle



This drawing explains visually the non-uniform parameterization. The points for negative x on the first drawing of the circle correspond to points far away from the vertex (tip) of the parabola (and thus far away from the origin, too). Because of the perspective foreshortening characteristic of projections, points which are (1) equidistant in 3D space but (2) placed progressively further away from the origin appear progressively clustered on the $w = 1$ plane.

Definitions: Circle

Can perceive projection of

$$\mathbf{q}(t) = (1 + t^2, 1 - t^2, 2t)$$

on plane $w = 1$ as finding Euclidean points of curve

$$\mathbf{r}(t) = (1 + t^2; 1 - t^2, 2t)$$

on the projective plane.

Definitions

A *rational curve* is a curve

$$\mathbf{p}(t) = (p_w(t); p_x(t), p_y(t))$$

on the projective plane where $p_w(t)$, $p_x(t)$, and $p_y(t)$ are polynomials.

Degree:

$$\max[\deg p_w(t), \deg p_x(t), \deg p_y(t)].$$

Projection:

$$\bar{\mathbf{p}}(t) = (p_x(t)/p_w(t), p_y(t)/p_w(t)),$$

provided $p_w(t) \neq 0$.

The class of rational curves contains all Bézier curves: the latter are just a special case where $p_w(t)$ is identically equal to one.

We will cover only planar rational curves, although, as usual, our discussion trivially extends to higher dimensional curves.

Specification

Specify rational curve

$$\bar{\mathbf{p}}(t) = (p_x(t)/p_w(t), p_y(t)/p_w(t))$$

by 3D Bézier curve

$$\mathbf{p}(t) = (p_w(t), p_x(t), p_y(t)),$$

called the *generating curve* of $\bar{\mathbf{p}}(t)$.

Specification

From Bézier points of $\mathbf{p}(t)$,

$$\mathbf{b}_i = \begin{pmatrix} w_i \\ x_i w_i \\ y_i w_i \end{pmatrix}, 1 \leq i \leq 3,$$

we get

$$\mathbf{p}(t) = \mathbf{b}_1 B_1(t) + \mathbf{b}_2 B_2(t) + \mathbf{b}_3 B_3(t),$$

which projects to

$$\bar{\mathbf{p}}(t) = \frac{w_1 \bar{\mathbf{b}}_1 B_1(t) + w_2 \bar{\mathbf{b}}_2 B_2(t) + w_3 \bar{\mathbf{b}}_3 B_3(t)}{w_1 B_1(t) + w_2 B_2(t) + w_3 B_3(t)},$$

where

$$\bar{\mathbf{b}}_i = (x_i, y_i).$$

We have assumed a degree 2 rational curve, but, as before, our discussion easily extends to curves of arbitrary degree. Note that we avoid the term "quadratic" for rational curves, as it may be mistaken as a reference to quadratic B-splines, a highly specialized case of degree 2 rational curves.

The Bernstein polynomials for quadratics are

$$B_1(t) = (1-t)^2 \quad B_2(t) = 2(1-t)t \quad B_3(t) = t^2,$$

since, as we stated in a past lecture, the Bernstein polynomials are the summands in the expansion of

$$[(1-t)+t]^2.$$

The reason that the Bézier points of $\mathbf{p}(t)$ are specified in this strange way is that the given form,

$$\mathbf{b}_i = \begin{pmatrix} w_i \\ x_i w_i \\ y_i w_i \end{pmatrix},$$

allows us to find the projection of the Bézier point $\mathbf{b}_i$ by inspection:

$$\bar{\mathbf{b}}_i = \begin{pmatrix} x_i w_i / w_i \\ y_i w_i / w_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}.$$

Of course, we have made the implicit assumption that $w_i \neq 0$; that is, no Bézier point can be a point-at-infinity. This assumption is ubiquitous in implementations of rational curves (usually, the stronger condition $w_i > 0$ is used). We will consider what happens when w_i is zero during the next lecture.

The derivation of the above formula for $\bar{\mathbf{p}}(t)$ becomes clearer if we write $\mathbf{p}(t)$ as

$$\begin{aligned}
\mathbf{p}(t) &= \sum_{i=1}^{3} \begin{pmatrix} w_i \\ x_i w_i \\ y_i w_i \end{pmatrix} B_i(t) \\
&= \begin{pmatrix} w_1 B_1(t) + w_2 B_2(t) + w_3 B_3(t) \\ w_1 x_1 B_1(t) + w_2 x_2 B_2(t) + w_3 x_3 B_3(t) \\ w_1 y_1 B_1(t) + w_2 y_2 B_2(t) + w_3 y_3 B_3(t) \end{pmatrix}.
\end{aligned}$$

Denoting the result by

$$\begin{pmatrix} w \\ xw \\ yw \end{pmatrix},$$

we immediately conclude that

$$\begin{aligned}
w &= w_1 B_1(t) + w_2 B_2(t) + w_3 B_3(t) \\
x &= \frac{1}{w}[w_1 x_1 B_1(t) + w_2 x_2 B_2(t) + w_3 x_3 B_3(t)] \\
y &= \frac{1}{w}[w_1 y_1 B_1(t) + w_2 y_2 B_2(t) + w_3 y_3 B_3(t)].
\end{aligned}$$

Specification: Bernstein Form

Bernstein form of degree 2 rational curve:

$$\bar{\mathbf{p}}(t) = \frac{w_1\bar{\mathbf{b}}_1 B_1(t) + w_2\bar{\mathbf{b}}_2 B_2(t) + w_3\bar{\mathbf{b}}_3 B_3(t)}{w_1 B_1(t) + w_2 B_2(t) + w_3 B_3(t)},$$

where

- the $\bar{\mathbf{b}}_i$'s are the *Bézier points* of the rational curve, and

- the w_i's are their *weights*.

For the Bézier "points" to be real points on the plane, their coefficients must sum to unity, which indeed they do (by inspection).

If all the weights are equal to one, then we get regular planar Bézier curves, as we remarked earlier.

Drawing

Can draw generating curve using polar interpolations and project on $w = 1$ plane: unintuitive.

Instead, do *rational* interpolations in 2D: *rational de Casteljau algorithm*.

Rational interpolation between two points: given

$$\begin{array}{ll} w_1 & (x_1, y_1) \\ w_2 & (x_2, y_2), \end{array}$$

result is

$$\begin{aligned} w &= w_1\lambda + w_2(1-\lambda) \\ \begin{pmatrix} x \\ y \end{pmatrix} &= \frac{1}{w}\left[w_1\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}\lambda + w_2\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(1-\lambda)\right]. \end{aligned}$$

The expressions for rational interpolation are derived from the corresponding linear interpolations on the 3D generating curve:

$$\begin{pmatrix} w \\ wx \\ wy \end{pmatrix} = \begin{pmatrix} w_1 \\ w_1 x_1 \\ w_1 y_1 \end{pmatrix} \lambda + \begin{pmatrix} w_2 \\ w_2 x_2 \\ w_2 y_2 \end{pmatrix} (1 - \lambda).$$

For those of you who are familiar with hardware texture mapping, you might recall rational interpolations: they are performed across triangles in order to figure out how a texture in object space projects onto the screen.

Drawing: Example

Let $\lambda = 0.5$ and

$$w_1 = 1 \qquad (x_1, y_1) = (2, 4)$$
$$w_2 = 3 \qquad (x_2, y_2) = (4, 6).$$

Interpolating in 3D, we obtain

$$\begin{pmatrix} w \\ wx \\ wy \end{pmatrix} = 0.5 \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} + 0.5 \begin{pmatrix} 3 \\ 12 \\ 18 \end{pmatrix} = \begin{pmatrix} 2 \\ 7 \\ 11 \end{pmatrix},$$

and hence

$$w = 2 \qquad (x, y) = (\frac{7}{2}, \frac{11}{2}).$$

Using rational interpolation, we get

$$\begin{aligned} w &= 1 \cdot 0.5 + 3(1 - 0.5) = 2 \\ \begin{pmatrix} x \\ y \end{pmatrix} &= \frac{1}{2} \left[1 \begin{pmatrix} 2 \\ 4 \end{pmatrix} 0.5 + 3 \begin{pmatrix} 4 \\ 6 \end{pmatrix} (1 - 0.5) \right] \\ &= \begin{pmatrix} 7/2 \\ 11/2 \end{pmatrix}. \end{aligned}$$

Drawing: Weights

Let $\lambda = 0.5$ and

$$\begin{aligned} w_1 &= 1 \qquad (x_1, y_1) = (2, 4) \\ w_2 &= e \qquad (x_2, y_2) = (4, 6). \end{aligned}$$

Using rational interpolation, we get

$$\begin{aligned} w &= 1 \cdot 0.5 + e(1 - 0.5) = 0.5(1 + e) \\ x &= \frac{2}{1+e}[1 \cdot 2 \cdot 0.5 + e4(1 - 0.5)] \\ &= 2\frac{1+2e}{1+e} = 2\frac{1/e+2}{1/e+1} \\ y &= \frac{2}{1+e}[1 \cdot 4 \cdot 0.5 + e6(1 - 0.5)] \\ &= 2\frac{2+3e}{1+e} = 2\frac{2/e+3}{1/e+1}. \end{aligned}$$

As $e \to \infty$, $1/e \to 0$.

So, as weight e increases, (x, y) moves closer to $(x_2, y_2) = (4, 6)$, regardless of λ.

This is why e is called the *weight* of a point.

Standard Form

Standard form: first and last weight are unity.

Let

$$\begin{aligned} w_1 &= 1 \qquad (x_1, y_1) = (1, 0) \\ w_2 &= e \qquad (x_2, y_2) = (0, 0) \\ w_3 &= 1 \qquad (x_3, y_3) = (0, 1). \end{aligned}$$

Then

$$\begin{aligned} \mathbf{p}(t) &= ((1-t)^2 + 2et(1-t) + t^2; (1-t)^2, t^2) \\ &= (2t^2(1-e) - 2t(1-e) + 1; (1-t)^2, t^2), \end{aligned}$$

and $p_w(t) = 0$ iff

$$\begin{aligned} t &= \frac{2(1-e) \pm \sqrt{4(1-e)^2 - 8(1-e)}}{4(1-e)} \\ &= \frac{(1-e) \pm \sqrt{e^2-1}}{2(1-e)}. \end{aligned}$$

This definition of standard forms applies to arbitrary degrees, not just degree 2 rational curves.

We found $\mathbf{p}(t)$ using the Bernstein form of quadratic 3D Bézier curves. Then, we rearranged the terms to express $p_w(t)$ as a quadratic in t. From this last expression, we just used the well known formula for roots of quadratic polynomials in order to find the roots of $p_w(t)$.

Classification

$\mathbf{p}(t)$ is parabola implies $\bar{\mathbf{p}}(t)$ is conic section. Which kind?

$\bar{\mathbf{p}}(t)$ goes to infinity when $p_w(t) = 0$ which happens when

$$t = \frac{(1-e) \pm \sqrt{e^2 - 1}}{2(1-e)}.$$

So:

- $|e| > 1$: $\bar{\mathbf{p}}(t)$ goes to infinity twice, so it is a hyperbola.

- $|e| = 1$: $\bar{\mathbf{p}}(t)$ goes to infinity once, so it is a parabola.

- $|e| < 1$: $\bar{\mathbf{p}}(t)$ never goes to infinity, so it is an ellipse.

Actually, conic section are frequently *defined* as projections of a generating parabola.

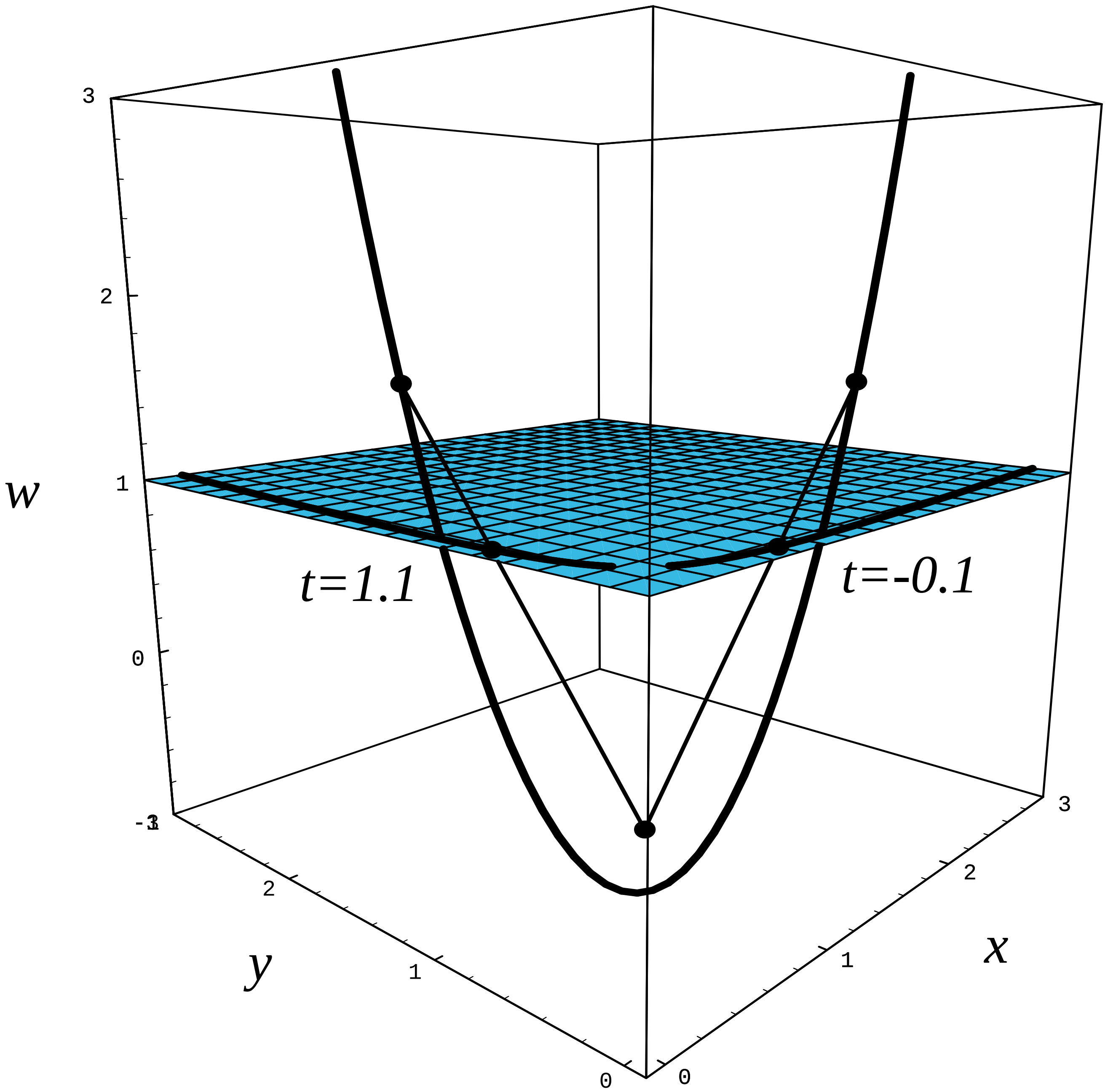
It is from projective geometry that we know that

- a hyperbola contains two points-at-infinity, e.g. $xy = 1$, or rather $xy = w^2$, contains the points-at-infinity $(0; 0, \lambda)$ and $(0; \lambda, 0)$,

- a parabola contains one point-at-infinity, e.g. $y = x^2$, or rather $yw = x$, contains the point-at-infinity $(0; 0, \lambda)$, and

- an ellipse does not contain any points-at-infinity.

Hyperbola ($e = -2$)



This plot, and six more that follow, depict the rational curve $\bar{\mathbf{p}}(t)$ and its generating curve $\mathbf{p}(t)$ for several values of e. To prevent cluttering up the plots, we only mark the origin, and two or three select points on the generating curve, as well as their projections. In particular, we chose not to mark any Bézier points since their locations can be easily deduced from their coordinates; it turns out that

- the first and third Bézier points of both $\mathbf{p}(t)$ and $\bar{\mathbf{p}}(t)$ appear where $\mathbf{p}(t)$, $\bar{\mathbf{p}}(t)$, and the plane $w = 1$ all intersect, while

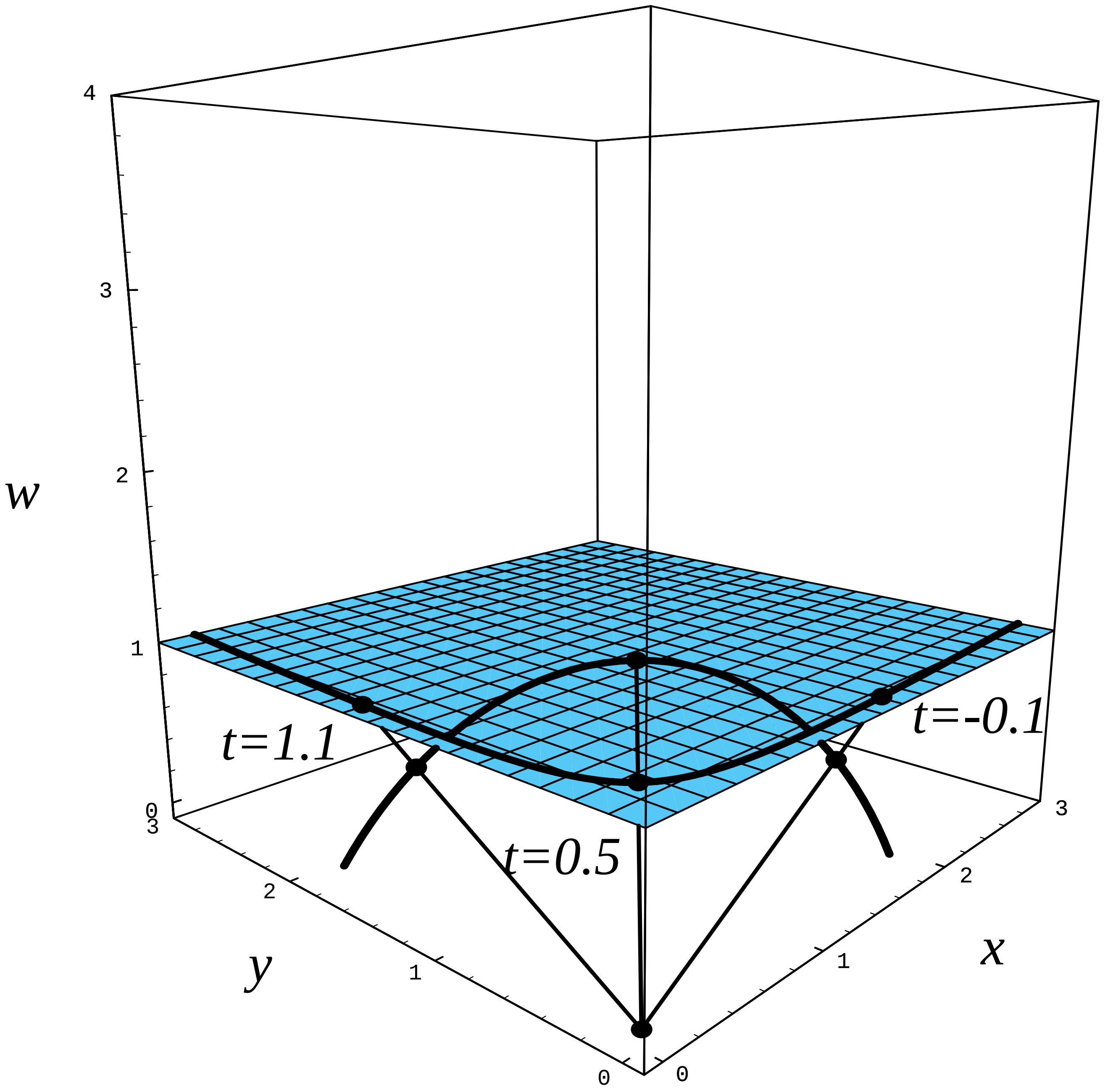
- the second Bézier point of $\mathbf{p}(t)$ is near the vertex of the parabola, close to the nearest edge of the box bounding the plot. Also, the second Bézier point of $\bar{\mathbf{p}}(t)$ is on the $w = 1$ plane, very near its intersection with the nearest edge of the bounding box.

Using as a guide the marked points, as well as the Bézier points (which correspond to times $t = 0$ and $t = 1$), trace the movement of a particle and its projection over time. This should give you some insight on the correspondence between the generating curve and its projection.

In this plot, the particle enters the field of view from the top right corner at some large negative value of t. As the particle moves towards the vertex of the parabola (which it reaches at $t = 0.5$), its projection moves from the center of the plot towards the right edge. Then, as time takes higher positive values, the particle climbs up the left branch of the parabola while its projection enters the plot from the left tip of the hyperbola and moves towards the center of the plot.

The missing section of the hyperbola (and similar missing sections from subsequent plots) is due to the fact that we only plot values of t between -1 and 2 or similar small ranges.

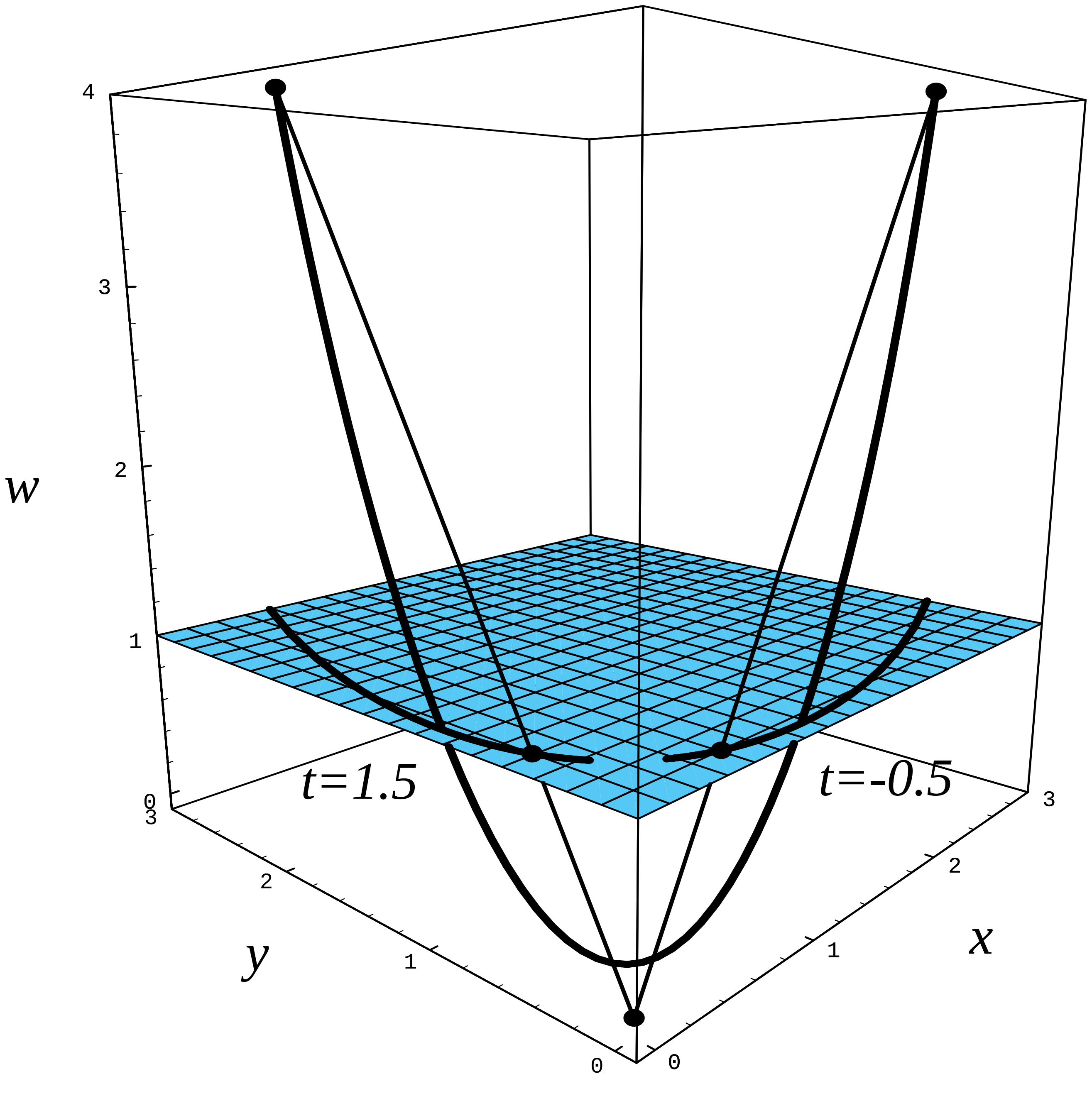
Hyperbola ($e = 2$)



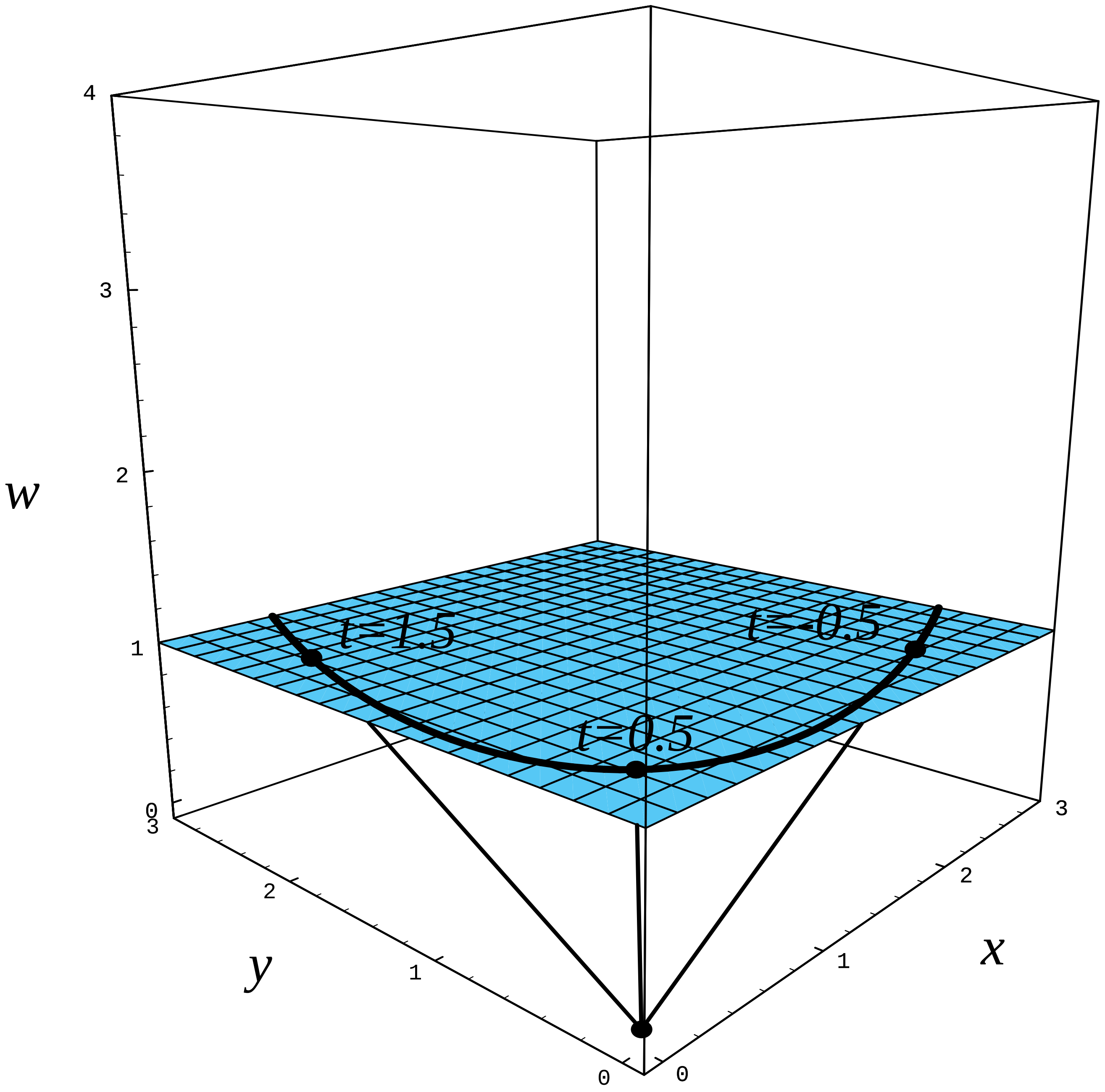
Observe that we get the same curve as on the previous slide, but with a different parameterization. We will address that issue shortly.

The point $\bar{\mathbf{p}}(0.5)$ is called the *shoulder point* of the curve. On the previous plot, $\bar{\mathbf{p}}(0.5)$ was a point-at-infinity, and hence it could not be plotted.

Parabola ($e = -1$)

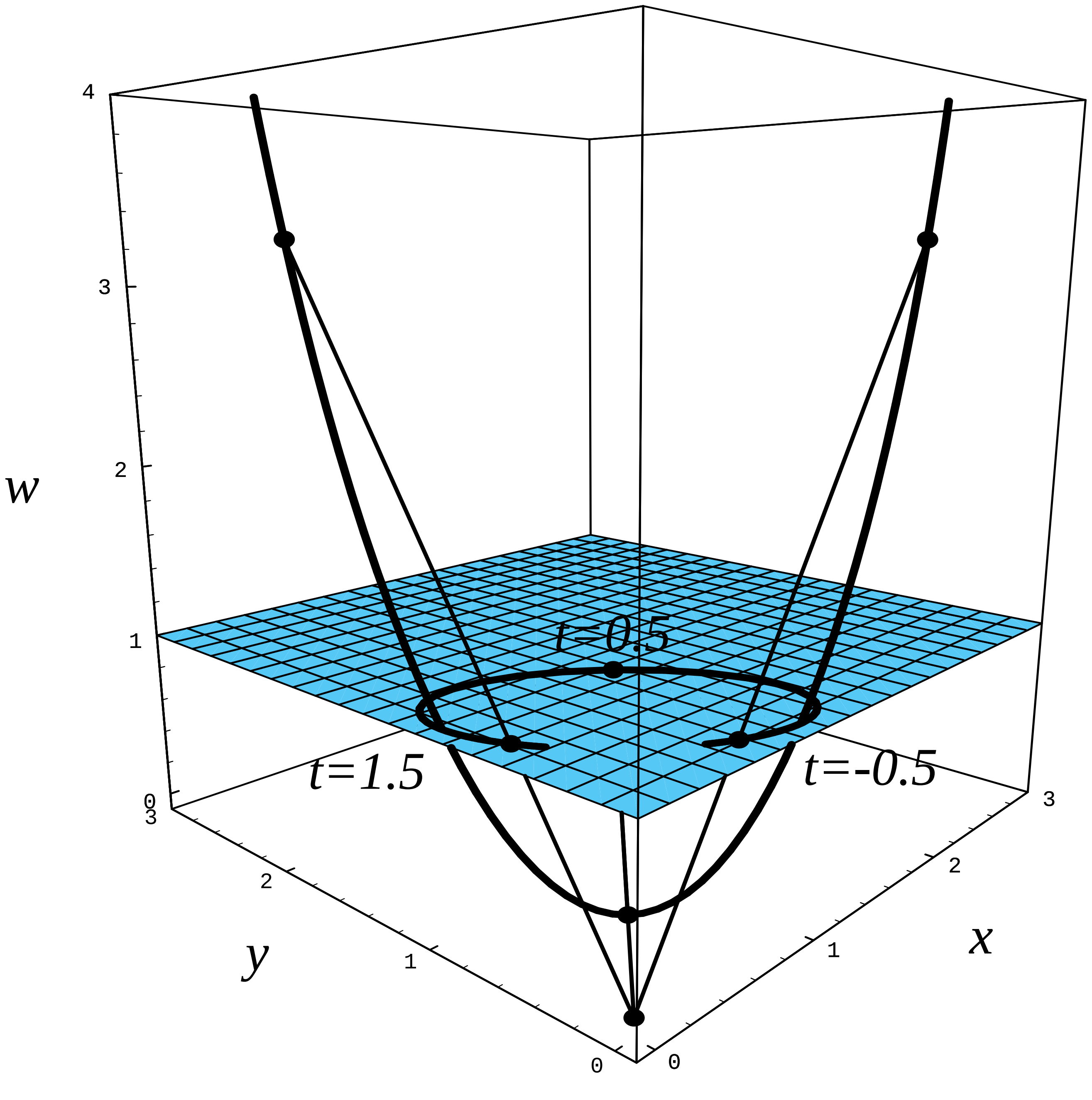


Parabola ($e = 1$)

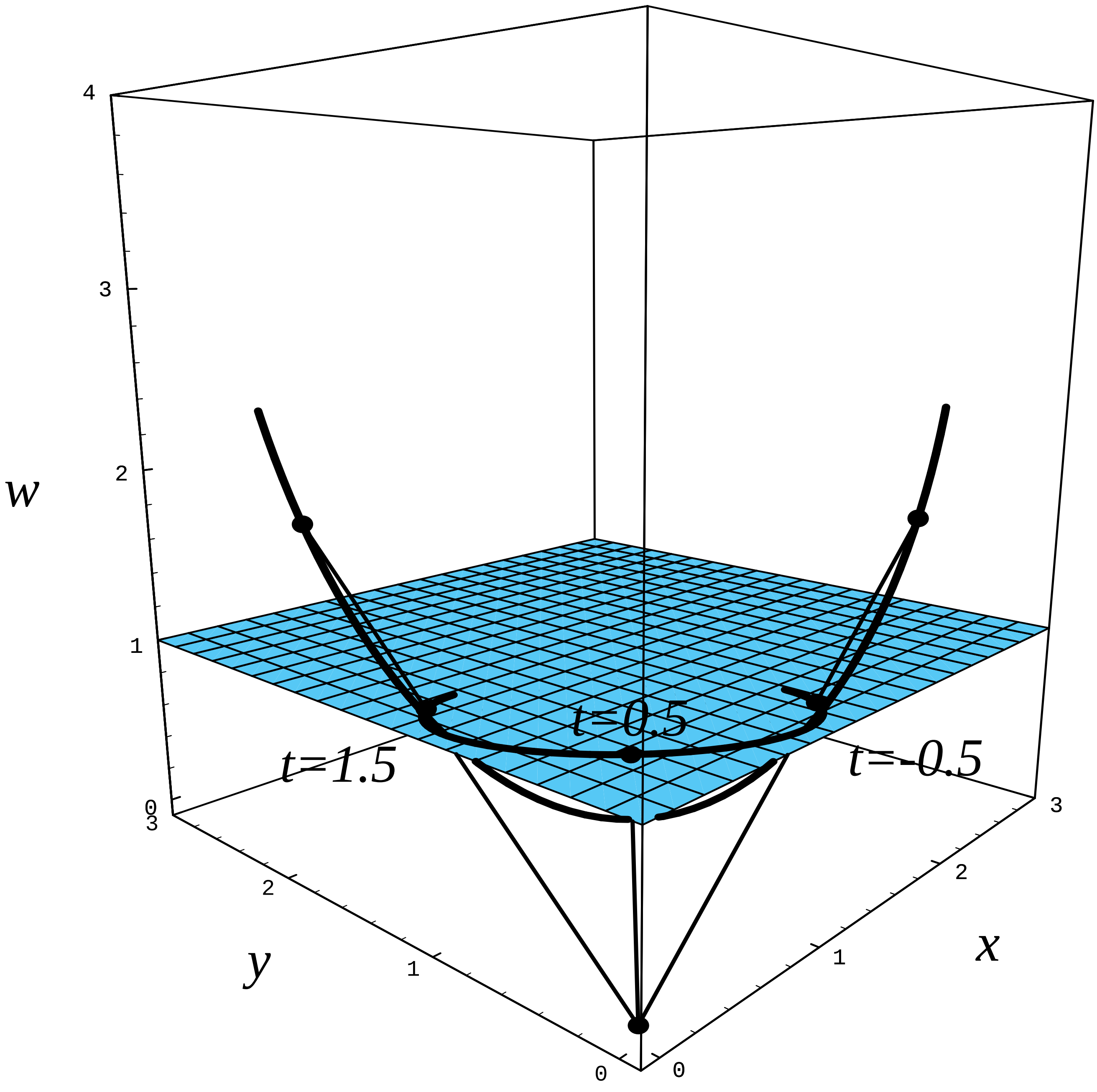


When $e = 1$, all the weights are unity, and thus the generating curve lies completely on the $w = 1$ plane, overlapping its projection.

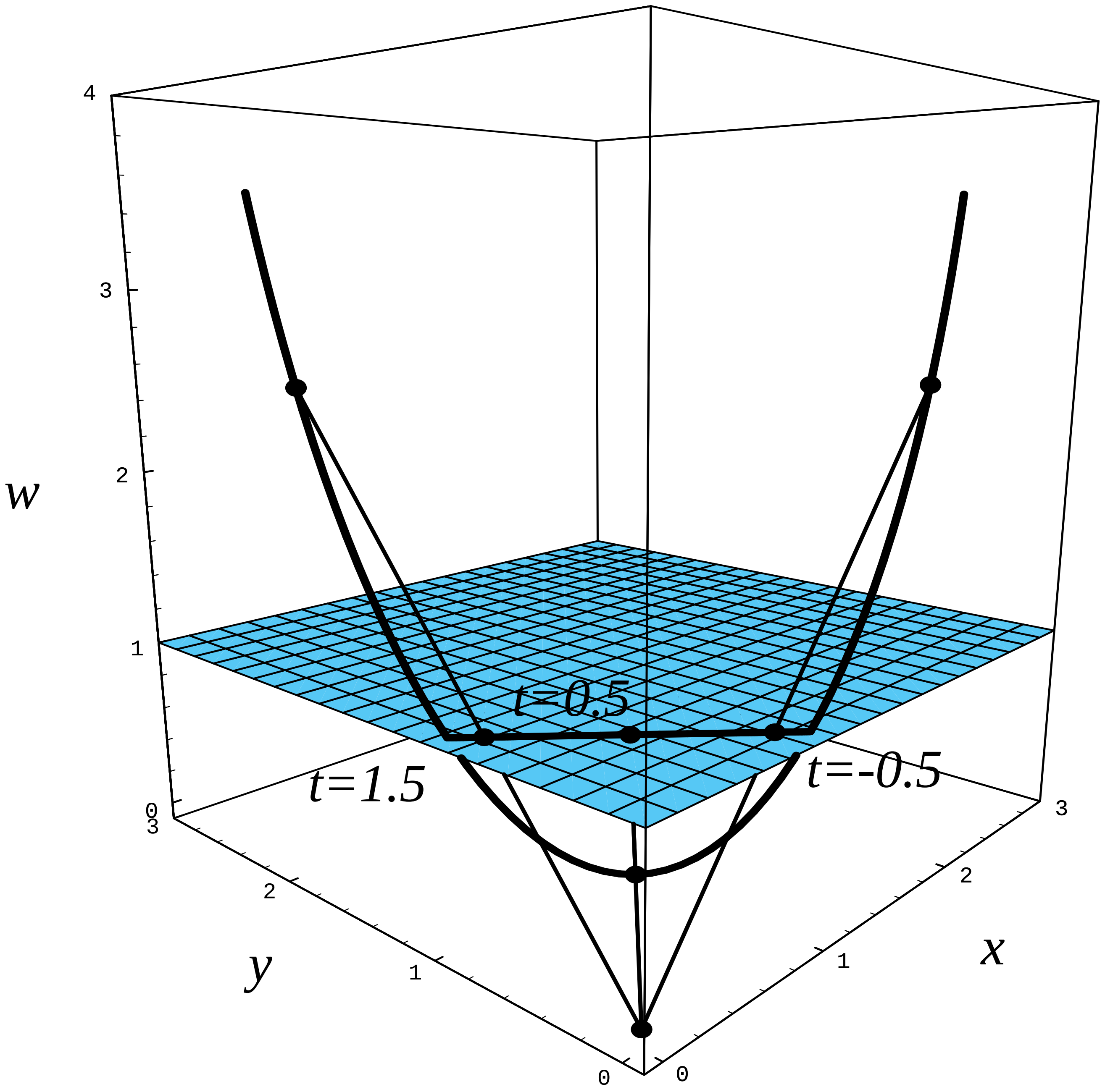
Ellipse ($e = -0.5$)



Ellipse ($e = 0.5$)



Line ($e = 0$)



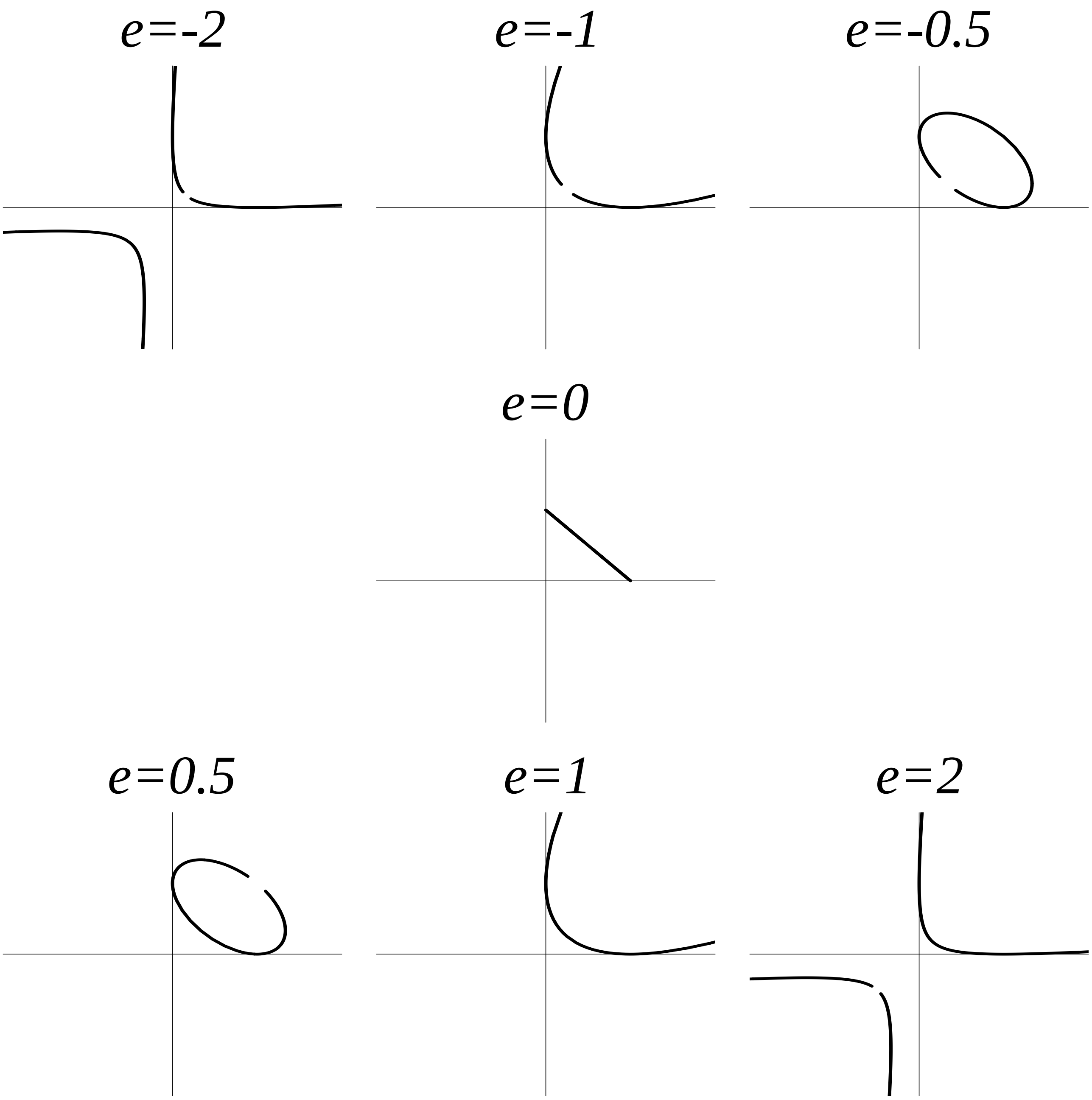
Examining the two previous plots carefully, we observe that the shoulder point and the point for $t \to \pm\infty$ have traded places when e flipped sign. What do we get right in the middle, when $e = 0$? The points for $t = 0.5$ and $t \to \pm\infty$ overlap and thus we get a straight line (or a *really* narrow, degenerate ellipse).

Another way to see this effect is a thought experiment. Imagine that you are holding a parabola, with its vertex right on your Adam's apple, and its two arcs aligned with your arms. Also, put a point light on your back and between your shoulders, and examine the shadow of the parabola on the wall in front of you. Now, start lifting your arms, slowly, in unison. When your arms are below or above the horizontal, the shadow has the shape of some conic section. But when your arms are completely horizontal, the shadow is a line.

As a minor remark, observe that we referred above to "the point for $t \to \pm\infty$", rather than "the points for $t \to \pm\infty$". The reason we can say this is that the generating curve is a parabola, and thus the limiting point is the same for both $t \to \infty$ and $t \to -\infty$.

Finally, $e = 0$ implies that the second Bézier point is a point-at-infinity. In this case, as we noted earlier, the Bernstein form of a rational curve is invalid; the projective form, however, is still valid, and this is what we used to generate this plot.

Degree 2 Rational Curves



Degree 2 Rational Curves

Remarks:

- All curves $\bar{\mathbf{p}}(t)$ interpolate $\bar{\mathbf{b}}_1 = (1, 0)$ and $\bar{\mathbf{b}}_3 = (0, 1)$.

- For all curves, tangents at $\bar{\mathbf{b}}_1$ and $\bar{\mathbf{b}}_3$ lie along lines joining corresponding point and $\bar{\mathbf{b}}_2 = (0, 0)$.

- Shoulder point always lies along line $(1, 1)$. Larger e draws shoulder point closer to origin.

- $\pm e$ both produce same curve.

Comments:

- This is to be expected since (1) the parabola $\mathbf{p}(t)$, as all Bézier curves, interpolates its own Bézier points $\mathbf{b}_1$ and $\mathbf{b}_3$ and (2) the $\bar{\mathbf{b}}_i$'s is the projections of the $\mathbf{b}_i$'s.

- This is to be expected since projective maps do not affect tangencies, and the tangents of the parabola $\mathbf{p}(t)$ lie along $\mathbf{b}_1\mathbf{b}_2$ at $\mathbf{b}_1$, and along $\mathbf{b}_3\mathbf{b}_2$ at $\mathbf{b}_3$.

- For very large negative values of e, the shoulder point is pushed so far away that it shows up on the all-negative quadrant.

- In other words, the absolute value of e is all that matters. Why? This is our next topic...

Reparameterization

What is $\mathbf{p}(t)$ for $-e$?

$$\mathbf{p}(t) = ((1-t)^2 - 2et(1-t) + t^2; (1-t)^2, t^2).$$

If I let

$$t = \frac{s}{2s-1},$$

I get

$$\begin{aligned}
\mathbf{p}(s) &= \begin{pmatrix} (2s-1-s)^2 - 2es(2s-1-s) + s^2 \\ (2s-1-s)^2 \\ s^2 \end{pmatrix} \\
&= ((s-1)^2 - 2es(s-1) + s^2; (s-1)^2, s^2) \\
&= ((1-s)^2 + 2es(1-s) + s^2; (1-s)^2, s^2),
\end{aligned}$$

which is the same curve!

We got the top equation from the general form of $\mathbf{p}(t)$ for the given weights:

$$\mathbf{p}(t) = ((1-t)^2 + 2et(1-t) + t^2; (1-t)^2, t^2).$$

All we did, was replace e by $-e$.

The algebra is easier than it seems: in all the terms of $\mathbf{p}(t)$, the sum of the powers of t and $1-t$ is 2, and so the denominators $(2s-1)^2$ cancel out immediately.

Weight Redundancy

$$\bar{\mathbf{p}}(t) = \frac{w_1 \bar{\mathbf{b}}_1 B_1(t) + w_2 \bar{\mathbf{b}}_2 B_2(t) + w_3 \bar{\mathbf{b}}_3 B_3(t)}{w_1 B_1(t) + w_2 B_2(t) + w_3 B_3(t)}$$

does not change if, for $\sigma \neq 0$,

$$w_1 \Rightarrow \sigma w_1 \quad w_2 \Rightarrow \sigma w_2 \quad w_3 \Rightarrow \sigma w_3.$$

Also, if, for $\rho \neq 0$,

$$t = s \frac{1}{\rho(1-s)+s},$$

then

$$1 - t = (1-s)\frac{\rho}{\rho(1-s)+s},$$

and

$$\bar{\mathbf{p}}(s) = \frac{\rho^2 w_1 \bar{\mathbf{b}}_1 B_1(s) + \rho w_2 \bar{\mathbf{b}}_2 B_2(s) + w_3 \bar{\mathbf{b}}_3 B_3(s)}{\rho^2 w_1 B_1(s) + \rho w_2 B_2(s) + w_3 B_3(s)}.$$

So same shape, but reparameterized, if

$$w_1 \Rightarrow \rho^2 w_1 \quad w_2 \Rightarrow \rho w_2 \quad w_3 \Rightarrow w_3.$$

The $\Rightarrow$ symbol denotes substitution; it should not be mistaken for a limit evaluation ($\rightarrow$).

The first redundancy follows by inspection: the scaling factor σ simply cancels out.

We can see the first redundancy on our first 3D slide, showing the parabola projecting onto a circle: all the parabolas which appear on the projection surface are scaled versions of one another, and they all project to identical points on the $w = 1$ plane.

Finding the result after substituting s for t is easy. Because t and $1 - t$ are respectively replaced by multiples of s and $1 - s$, the Bernstein polynomials $B_i(t)$ turn into the corresponding $B_i(s)$. Also, because in every Bernstein polynomial the sum of the powers of t and $1 - t$ is 2, the denominators $[\rho(1 - s) + s]^2$ cancel out. All that's left is the powers of ρ, shown above.

It follows that the three weights only allow us one degree of freedom in manipulating the shape of a degree 2 rational curve. Hence, every such curve falls under our earlier analysis of the standard form of degree 2 rational curves. Moreover, our observation that the sign of e did not alter the shape of the curve is a special instant of our last result: reparameterizing using $\rho = -1$ flips the weight of the second Bézier point.

Very similar results hold for higher dimensions, as well. Hence by scaling the weights and multiplying them by a geometric progression, we can always find the standard form of any rational curve — provided both the first and last weight are non-zero. In more detail, if w_1 and w_n are the first and last weights of a degree $d = n - 1$ rational curve, scaling all weights by $\sigma = 1/w_n$, and subsequently reparameterizing using

$$\rho = \sqrt[d]{\frac{w_n}{w_1}},$$

yields the standard form of the rational curve.

References

- Böhm: sections 9, 20, 28, 30.

- Farin: pages 113-114, 136, 183, 159–165, 233-239, 244-246, 251–252, 257-258.

Böhm is in the course reader, and Farin is on reserve at the MCS library.