More precise determination of the bounding boxes of `tikzpictures`

Current status

Ti*k*Z determines the bounding box of (cubic) Bezier curves by establishing the smallest rectangle that contains the end point and the two control points of the curve (cf. Figure 1).

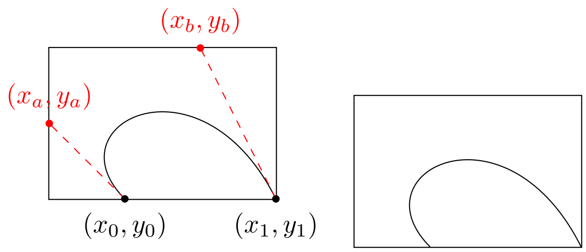

Figure 1: Ti*k*Z bounding boxes and an example.

This may lead to drastic overestimates of the bounding box (cf. Figure 2).

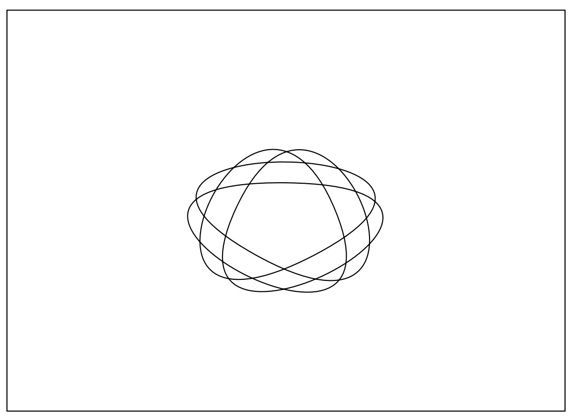

Figure 2: Example from `https://tex.stackexchange.com/q/43621/121799`.

Computing the bounding box

Establishing the precise bounding box has been discussed in various places, the following discussion uses in part the results from `https://pomax.github.io/bezierinfo/`. What is a cubic Bezier curve? A cubic Bezier curve running from (x_0, y_0) to (x_1, y_1) with control points (x_a, y_a) and (x_a, y_a) can be parametrized by

$$\gamma(t) \;=\; \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \;=\; \begin{pmatrix} t^3 x_1 + 3t^2(1-t)x_b + (1-t)^3 x_0 + 3t(1-t)^2 x_a \\ t^3 y_1 + 3t^2(1-t)y_b + (1-t)^3 y_0 + 3t(1-t)^2 y_a \end{pmatrix} , \tag{1}$$

where t runs from 0 to 1 (and $\gamma(0) = (x_0, y_0)$ and $\gamma(1) = (x_1, y_1)$). Surely, the bounding box has to contain (x_0, y_0) and (x_1, y_1). If the functions $x(t)$ and $y(t)$ have extrema in the interval $[0, 1]$, then the bounding box will in general be larger than that. In order to determine the extrema of the curve, all we need to find the extrema of the functions $x(t)$ and $y(t)$ for $0 \le t \le 1$. That is, we need to find the solutions of the quadratic equations

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) \;=\; 0 \quad \text{and} \quad \frac{\mathrm{d}y}{\mathrm{d}t}(t) \;=\; 0\,. \tag{2}$$

Let's discuss x, y is analogous. If the discriminant

$$d \;:=\; (x_a - x_b)^2 + (x_1 - x_b)(x_0 - x_a) \tag{3}$$

is greater than 0, there are two solutions

$$t_\pm \;=\; \frac{x_0 - 2x_a + x_b \pm \sqrt{d}}{x_0 - x_1 - 3(x_a - x_b)}\,. \tag{4}$$

In this case, we need to make sure that the bounding box contains, say $(x(t_-), y_0)$ and $(x(t_+), y_0)$. If $d \le 0$, the bounding box does not need to be increased in the x direction. One can plug $t_\pm$ back into (1), this yields

$$x_- \;=\; \frac{x_0^2 x_1 + x_0 x_1^2 - 3x_0 x_1 x_a + 6x_1 x_a^2 + 2x_a^3 - 3(x_0 + x_a)(x_1 + x_a)x_b + 3(2x_0 - x_a)x_b^2 + 2x_b^3 - 2\sqrt{d}(x_0 x_1 - x}{(x_0 - x_1 - 3x_a + 3x_b)^2} \tag{5a}$$

$$x_+ \;=\; \frac{x_0^2 x_1 + x_0 x_1^2 - 3x_0 x_1 x_a + 6x_1 x_a^2 + 2x_a^3 - 3(x_0 + x_a)(x_1 + x_a)x_b + 3(2x_0 - x_a)x_b^2 + 2x_b^3 + 2\sqrt{d}(x_0 x_1 - x}{(x_0 - x_1 - 3x_a + 3x_b)^2} \tag{5b}$$

As already mentioned, the analogous statements apply to $y(t)$.

It is rather straightforward to implement this prodecure in Ti*k*Z. The relevant macros are `\pgf@lt@curveto` (and `\pgf@nlt@curveto`).[1] The macro `\pgf@lt@curveto` takes six arguments, which are x_a, y_a, x_b, y_b, x_1 and y_1 (in that order). x_0 and y_0 are stored in `\pgf@path@lastx` and `\pgf@path@lastx`, respectively.

[1] Some care has to be taken when squaring lengths since they are all measured in points, and the square can easily become large and trigger a `dimension too large` error. When computing the discriminant d, I thus divided these distances by some taming factor that I derived from the input values, and this seems to work in the tests.

Examples

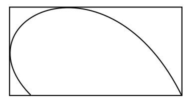

Figure 3: Tight bounding box for figure 1.

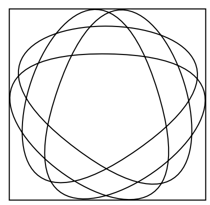

Figure 4: Tight bounding box for figure 2.