B-Spline Geometry

Chad B. Hovey[1]

Sandia Injury Biomechanics Laboratory
Sandia National Laboratories[2]

2021-12-15

[1]Author e-mail: chovey@sandia.gov

[2]The Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Contents

Chapter 1

Introduction

1.1 Parameter Space

In Bézier geometry, parameter space t for curves[1] is a real number between zero and unity, inclusive,

$$t \in \mathbb{R} \subset [\,0.0,\ 1.0\,]. \tag{1.1}$$

For B-spline geometry, the parameter space is taken as a real number between zero and some number, $\mathsf{T}_{\mathtt{I}}$, typically larger than unity as seen in the forthcoming discussion. For

[1]This is extended by u for s$\underline{\text{u}}$rfaces and again by v for $\underline{\text{v}}$olumes.

now, we say

$$t \in \mathbb{R} \subset [\ \mathsf{T}_0,\ \mathsf{T}_{\mathtt{I}}\ ]. \tag{1.2}$$

So, the parameter space for Bézier curves will be a special case of the parameter space for B-spline curves when $\mathsf{T}_0 = 0.0$ and $\mathsf{T}_{\mathtt{I}} = 1.0$.

1.2 Knots, Knot Spans, Knot Vectors

Next, we identify discrete, non-decreasing values along this interval $[\ \mathsf{T}_0,\ \mathsf{T}_{\mathtt{I}}\ ]$, and define these values as **knots**. **Knots** decompose the parameter space into sequential sub-intervals, called **knot spans**.[2] The set of $(\mathtt{I}+1)$ knots compose a **knot vector $\mathbf{\mathsf{T}}$**, *viz.*

$$\mathbf{\mathsf{T}} = \{\ \mathsf{T}_0, \mathsf{T}_1, \mathsf{T}_2, \ldots, \mathsf{T}_{\mathtt{I}}\ \} = \{\mathsf{T}_i\}_{i=0}^{\mathtt{I}} \tag{1.3}$$

Because knots mark the termination points, beginning and end, of knot spans, they impart a measure on the knot span, which is simply the difference between the values at sequential knots, and may be as small as zero, since knot sequence values are non-decreasing. For example, the value of the first knot span is equal to the value $(\mathsf{T}_1 - \mathsf{T}_0)$. A knot vector with $(\mathtt{I}+1)$ knots has $(\mathtt{I})$ knot spans.

[2] For curves, a knot in parameter space will get mapped to a point in physical space. For surfaces, a knot will get mapped to a curve. For volumes, a knot will get mapped to a surface. For now, consider only curves with knots.

Remark 1.2.1. Recastability of the Parameter Space
Since the B-Spline domain $[\,0.0,\ \mathsf{T}_{\mathtt{I}}\,]$ is a *parameter* space, it can be recast. Two examples follow:

- **Normalization**: The entire interval can be divided by $\mathsf{T}_{\mathtt{I}}$, making the new parameter space be $[\,0.0,\ 1.0\,]$, which is a recovery of the Bézier parameter space.
- **Offset**: The interval may be shifted up or down by some constant value. Thus, T_0 is not necessarily always zero.

Remark 1.2.2. Unit Knot Span Convention
It is a convention, but not a requirement, to denote knot values as non-negative *integer* values starting from zero, though they actually have non-negative *real* values. For example, the knot vector $\mathbf{T} = \{\ 0.0,\ 0.5,\ 1.0\ \}$ can be equally-well represented as $\mathbf{T} = \{\ 0.0,\ 1.0,\ 2.0\ \}$. Both have three knots but only the latter has a unit knot span. The unit knot span convention is used because it is often convenient to count knots, one by one.

Remark 1.2.3. Connection to Finite Element Analysis (FEA)
Knot spans will also be known as **elements** because we perform numerical quadrature over a knot span in isogeometric analysis (IGA) [Cottrell et al., 2009]. In IGA, the parent (or local or parameterized) element is the knot span. All of the knot spans described by a single knot vector are defined as a **patch**. A patch spans the B-spline parameter space.

In contrast, isoparametric analysis used for FEA has two notions of element: the parent (or local or parameterized) element and the physical (global) element.

1.3 Uniform Knot Vectors

When all the knot spans of a given knot vector are equal, the knot vector is **uniform**. Otherwise, the knot vector is **non-uniform**. We will begin the discussion with uniform knot vectors because they are the easier of the two variants to develop.

Remark 1.3.4. Knot Vectors Notation
Knot vectors are composed of real numbers. Hereafter we will write them without decimal values when possible. This shorthand notation should not be construed as integer values. Knot vectors belong to the set of real numbers and (generally) not to the set of integers.

Example 1.
A *uniform* knot vector containing 10 knots might be written as

$$\mathbf{T} = \{\, 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\,\},\ \text{with } t \in \mathbb{R} \subset [\,0,\ 9\,]. \tag{1.4}$$

□

Example 2.
A *non-uniform* knot vector containing 10 knots might be written as

$$\mathbf{T} = \{\, 0,\ 0,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\,\},\ \text{with } t \in \mathbb{R} \subset [\,0,\ 9\,]. \tag{1.5}$$

The first two knot spans have a value of zero and two, respectively. The remaining knot spans have a value of one. Thus, the knot vector is non-uniform. Notice also the repeated knot value of zero at the beginning of the knot vector. This is allowed since knots are a non-decreasing sequence. Repeated beginning and end knots will have a particular significance, as shown later in Section 1.5. □

1.4 Basis Functions

Let the B-spline normalized basis function of degree p be written N^p. Here, p denotes degree; it is not an exponent. After developing the basis functions, we then use them to construct B-spline curves in Chapter 2. The "B" in B-spline stands for **basis**.

The first normalized basis function is the unit piecewise constant, defined as

$$\text{for } p = 0: \quad N_i^0(t) \triangleq \begin{cases} 1 & \text{if } \mathsf{T}_i \leq t < \mathsf{T}_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.6}$$

Notice for the non-zero range, defined over $\mathsf{T}_i \leq t < \mathsf{T}_{i+1}$, the domain

- left-hand-side uses $\leq$, but the
- right-hand-side uses $<$ (and not $\leq$).

Example 3.
B-spline constant. Figure 1.1 shows $N_i^0(t)$ from (1.6), the unit piecewise constant basis

functions (degree $p = 0$), in parametric space, $t \in [\ \mathsf{T}_0,\ \mathsf{T}_{\mathtt{I}}\ ]$, $\mathtt{I} = 6$, for the uniform knot vector

$$\mathbf{\mathsf{T}} = \{\ \mathsf{T}_0, \mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4, \mathsf{T}_5, \mathsf{T}_6\ \} = \{\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\ \}. \tag{1.7}$$

$\square$

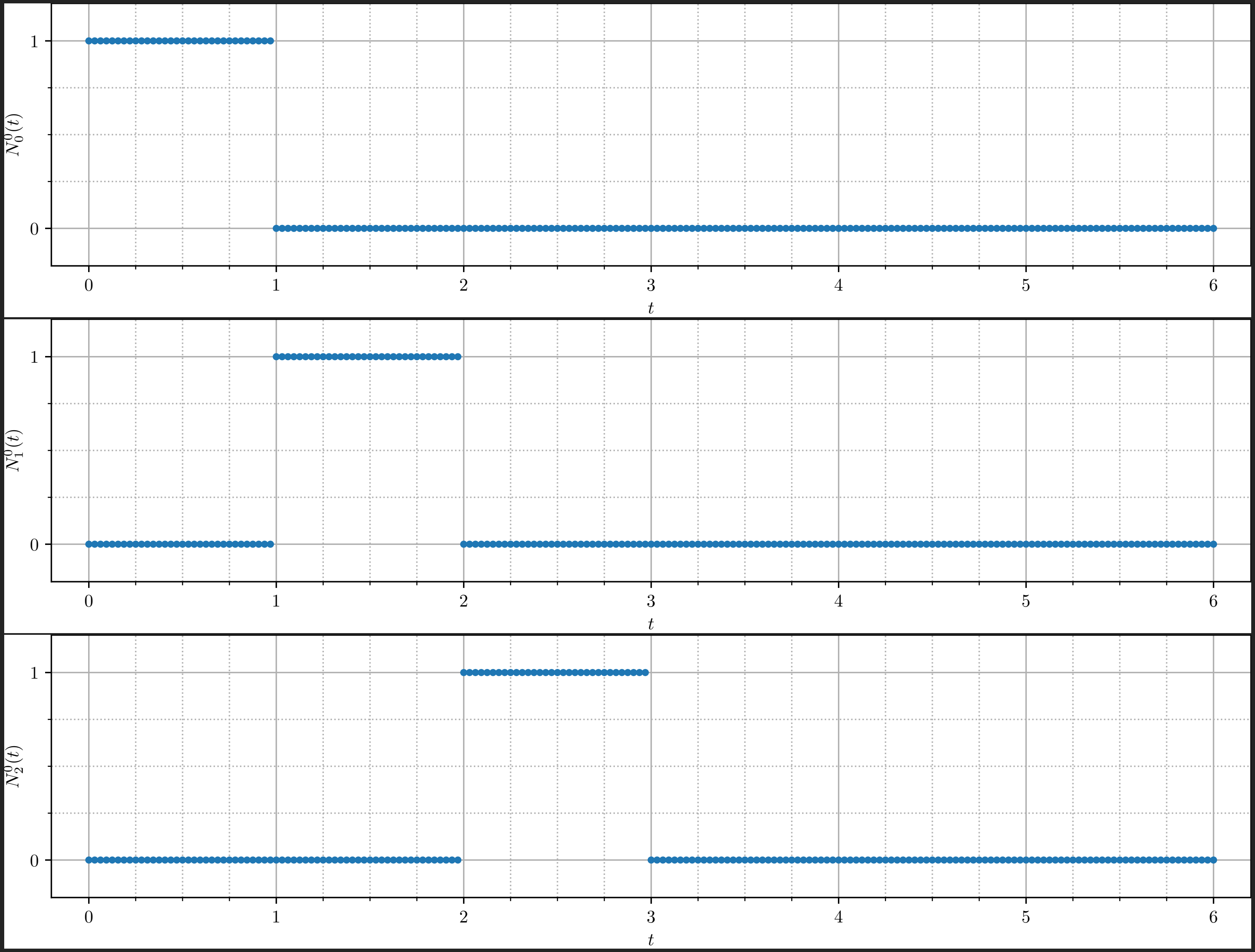

Figure 1.1: B-spline constant ($p = 0$) basis function. See `plot_bspline_basis_manual.py` on GitHub.

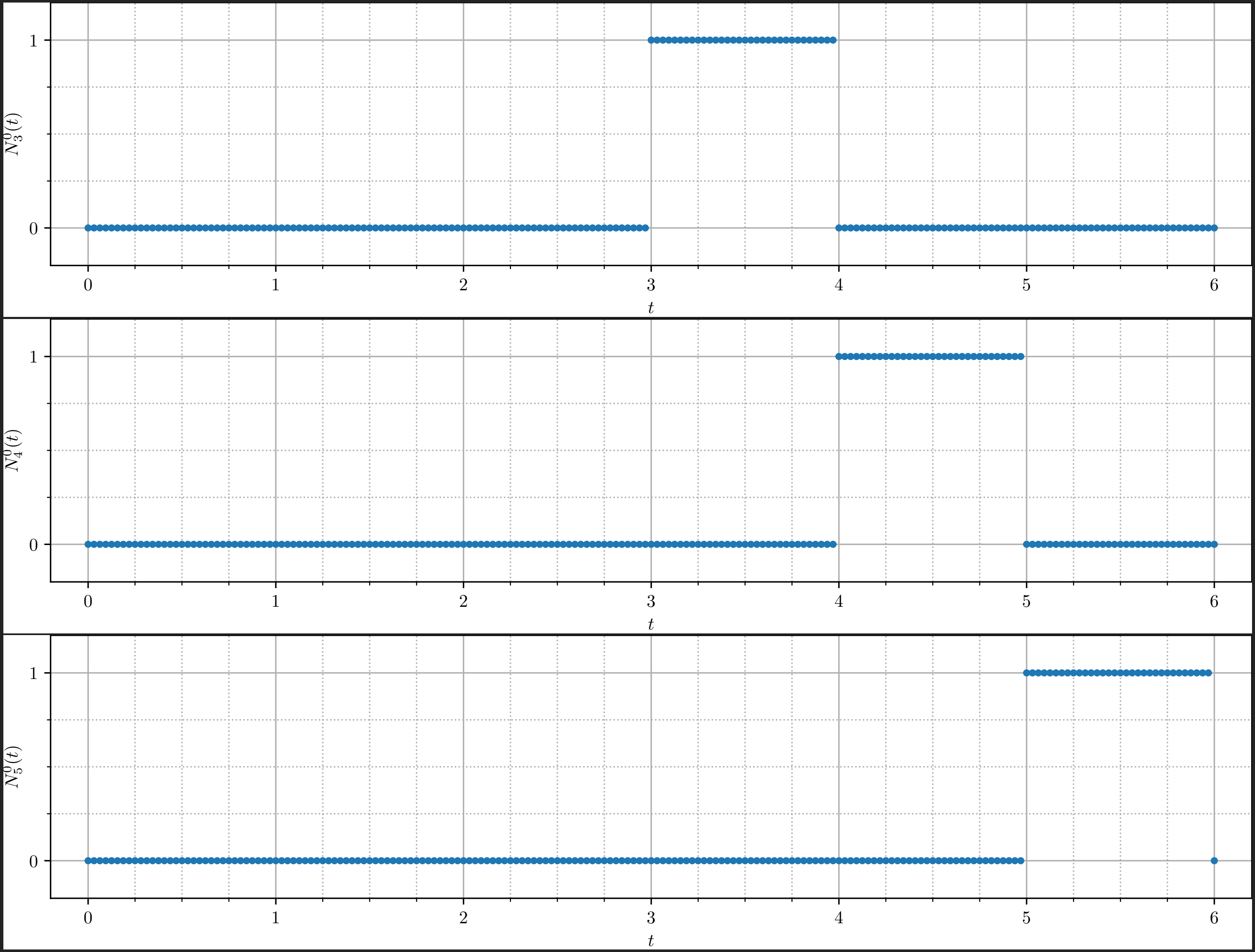

Figure 1.2: Continued from previous figure.

For a basis function of degree $p > 0$, *e.g.*, $p = 1, 2, 3, \ldots$, the normalized basis functions are defined by the **Cox-de Boor recursion formula**:

$$\text{for } p \geq 1: \quad N_i^p(t) \triangleq \frac{t - \mathsf{T}_i}{\mathsf{T}_{i+p} - \mathsf{T}_i} N_i^{p-1}(t) + \frac{\mathsf{T}_{i+p+1} - t}{\mathsf{T}_{i+p+1} - \mathsf{T}_{i+1}} N_{i+1}^{p-1}(t). \tag{1.8}$$

The recursive definition can lead to cases where $0/0$ is encountered. In these cases, the quotient is simplied defined as zero, *viz.*

$$\begin{aligned} &\texttt{if } \text{Eq. (1.8)} \Longrightarrow 0/0, \\ &\texttt{then } \text{Eq. (1.8)} \overset{\text{set}}{=} 0. \end{aligned} \tag{1.9}$$

Example 4.
B-spline linear. Using (1.8), the first $(i = 0)$ normalized basis function of degree $(p = 1)$ is

$$N_0^1(t) = \frac{t - \mathsf{T}_0}{\mathsf{T}_{0+1} - \mathsf{T}_0} N_0^{1-1}(t) + \frac{\mathsf{T}_{0+1+1} - t}{\mathsf{T}_{0+1+1} - \mathsf{T}_{0+1}} N_{0+1}^{1-1}(t), \tag{1.10}$$

$$= \frac{t - \mathsf{T}_0}{\mathsf{T}_1 - \mathsf{T}_0} N_0^0(t) + \frac{\mathsf{T}_2 - t}{\mathsf{T}_2 - \mathsf{T}_1} N_1^0(t). \tag{1.11}$$

Review of Figure 1.1 shows $N_0^0(t)$ and $N_1^0(t)$ act as “on” and “off” switches, since

$$N_0^0(t) = \begin{cases} 1 & \text{if } \mathsf{T}_0 \leq t < \mathsf{T}_1, \\ 0 & \text{otherwise;} \end{cases} \quad \text{and,} \quad N_1^0(t) = \begin{cases} 1 & \text{if } \mathsf{T}_1 \leq t < \mathsf{T}_2, \\ 0 & \text{otherwise.} \end{cases} \tag{1.12}$$

Thus,

$$N_0^1(t) = \begin{cases} (t - \mathsf{T}_0) \,/\, (\mathsf{T}_1 - \mathsf{T}_0) & \text{if } \mathsf{T}_0 \le t < \mathsf{T}_1, \\ (\mathsf{T}_2 - t) \,/\, (\mathsf{T}_2 - \mathsf{T}_1) & \text{if } \mathsf{T}_1 \le t < \mathsf{T}_2, \\ 0 & \text{otherwise.} \end{cases} \tag{1.13}$$

Figure 1.3 shows the B-spline linear basis functions (degree $p = 1$) over the same knot vector used for Figure 1.1. Note, for the given knot vector $\mathbf{T} = \{\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\ \}$, there is one fewer complete *linear* basis function than there is complete *constant* basis function. Explained below, this is due to local support, which increases with increasing degree and thus decreases the number of complete basis functions that can exist in the extents of the knot vector. □

Remark 1.4.5. Notice the first ($i = 0$) normalized basis function, $N_i^p(t)$, of degree p requires $(p+1)$ knots. Specifically with $p = 1$ in (1.13), $N_0^1(t)$ requires knots $\{\ \mathsf{T}_0, \mathsf{T}_1, \mathsf{T}_2\ \}$ to be defined. This will give rise to a relationship between the number of knots and both the degree and number of control points in (2.2).

Since we have not yet introduced control points, this concept is a bit ambiguous for now. However, at this early point, it is useful to seed the notion of the first basis function (which will eventually be multiplied by the first control point) of degree p requires $(p + 1)$ knots.

This example illustrates the pattern of **local support**. This pattern can be stated as follows:

A B-spline basis function of degree p
will have local support over $(p+1)$ knot spans.

Figure 1.3 also shows the pattern of **periodicity** in the basis functions for $N_i^1(t)$. In general, the unit piecewise linear (degree $p = 1$) basis function at knot T_i can be written as

$$N_i^1(t) = \begin{cases} (t - \mathsf{T}_i) \; / \; (\mathsf{T}_{i+1} - \mathsf{T}_i) & \text{if } \mathsf{T}_i \leq t < \mathsf{T}_{i+1}, \\ (\mathsf{T}_{i+2} - t) \; / \; (\mathsf{T}_{i+2} - \mathsf{T}_{i+1}) & \text{if } \mathsf{T}_{i+1} \leq t < \mathsf{T}_{i+2}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.14}$$

The periodicity pattern exists for all B-spline basis functions $N_i^p(t)$ of any degree $p \geq 0$.

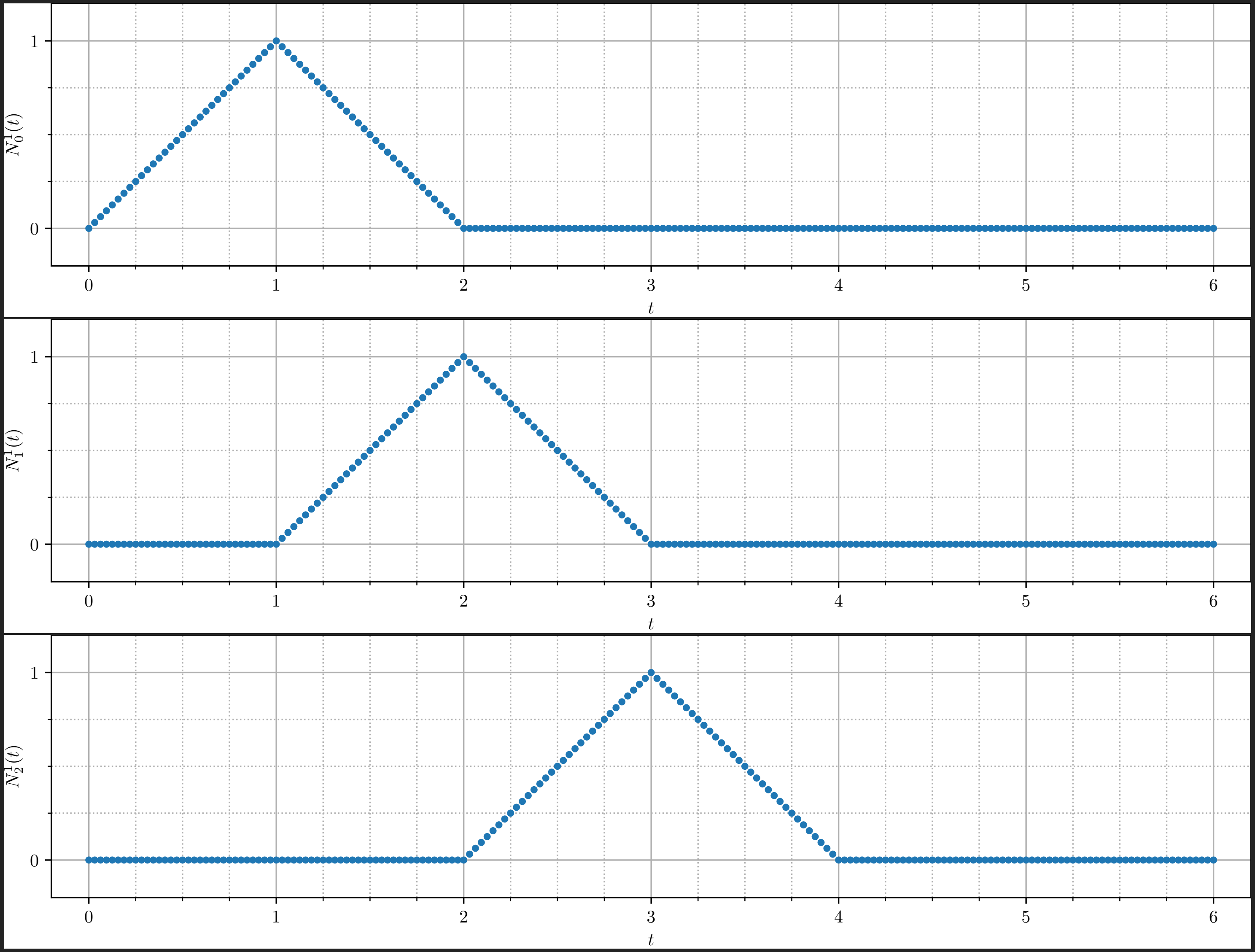

Figure 1.3: B-spline linear ($p = 1$) basis function. See `plot_bspline_basis_manual.py` on GitHub.

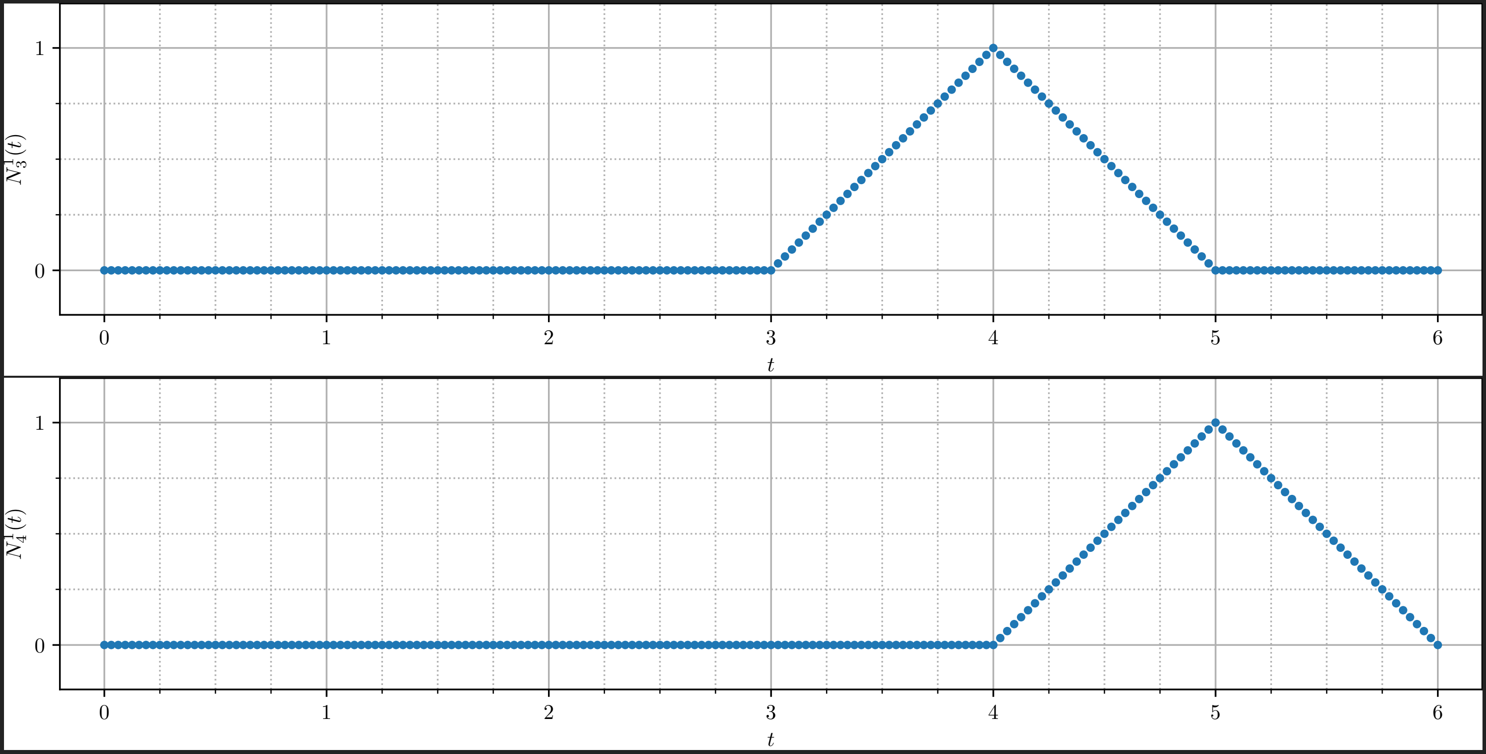

Figure 1.4: Continued from previous figure.

The local support property means that B-spline basis functions of increasing degree require an increasing number of knots to be defined. Increasing the degree of the B-spline basis tends to both increase the duration and decrease the amplitude of the non-zero values of the function. A basis function of degree p also depends on the basis functions of decreasing order, e.g., $(p-1)$, $(p-2)$, and so on. This dependence is defined through the Cox-de Boor relationships.

Figure 1.5 illustrates the Cox-de Boor recursion algorithm, with local support over knot spans.

$$
\begin{array}{ccccccccc}
\mathsf{T}_i & & \mathsf{T}_{i+1} & & \mathsf{T}_{i+2} & & \mathsf{T}_{i+3} & & \mathsf{T}_{i+4} \\
\vdots & \iddots & \vdots & \iddots & \vdots & \iddots & \vdots & \iddots & \\
N_i^0 & & N_{i+1}^0 & & N_{i+2}^0 & & N_{i+3}^0 & & \\
\vdots & \iddots & \vdots & \iddots & \vdots & \iddots & & & \\
N_i^1 & & N_{i+1}^1 & & N_{i+2}^1 & & & & \\
\vdots & \iddots & \vdots & \iddots & & & & & \\
N_i^2 & & N_{i+1}^2 & & & & & & \\
\vdots & \iddots & & & & & & & \\
N_i^3 & & & & & & & &
\end{array}
$$

Figure 1.5: Graphical illustration of Cox-de Boor recursion algorithm up to the degree of cubic ($p = 3$).

Figure 1.6 illustrates the Cox-de Boor recursion algorithm, with local support over knot spans, reimagined with a gridded shape.

$$\begin{tikzcd}
 & & & \mathsf{T}_i & & & & \\
\underline{\text{degree}} & \underline{\#\ \text{spans}} & \underline{\#\ \text{knots}} & & & & & \\
p=0 & 1 & 2 & N_i^0 \arrow[uu] \arrow[r] & \mathsf{T}_{i+1} & & & \\
 & & & & & & & \\
p=1 & 2 & 3 & N_i^1 \arrow[uu] \arrow[r] & N_{i+1}^0 \arrow[uu] \arrow[r] & \mathsf{T}_{i+2} & & \\
 & & & & & & & \\
p=2 & 3 & 4 & N_i^2 \arrow[uu] \arrow[r] & N_{i+1}^1 \arrow[uu] \arrow[r] & N_{i+2}^0 \arrow[uu] \arrow[r] & \mathsf{T}_{i+3} & \\
 & & & & & & & \\
p=3 & 4 & 5 & N_i^3 \arrow[uu] \arrow[r] & N_{i+1}^2 \arrow[uu] \arrow[r] & N_{i+2}^1 \arrow[uu] \arrow[r] & N_{i+3}^0 \arrow[uu] \arrow[r] & \mathsf{T}_{i+4}
\end{tikzcd}$$

Figure 1.6: Graphical illustration of Cox-de Boor recursion algorithm up to the degree of cubic ($p = 3$), with gridded arrangement.

Example 5.

B-spline quadratic. Using (1.8), the i^{th} normalized basis function of degree $(p = 2)$ is

$$\begin{aligned} N_i^2(t) &= \frac{t - \mathsf{T}_i}{\mathsf{T}_{i+2} - \mathsf{T}_i} N_i^1(t) + \frac{\mathsf{T}_{i+3} - t}{\mathsf{T}_{i+3} - \mathsf{T}_{i+1}} N_{i+1}^1(t), \qquad (1.15) \\ &= \frac{t - \mathsf{T}_i}{\mathsf{T}_{i+2} - \mathsf{T}_i} \left\{ \frac{t - \mathsf{T}_i}{\mathsf{T}_{i+1} - \mathsf{T}_i} N_i^0(t) + \frac{\mathsf{T}_{i+2} - t}{\mathsf{T}_{i+2} - \mathsf{T}_{i+1}} N_{i+1}^0(t) \right\} + \\ &\quad \frac{\mathsf{T}_{i+3} - t}{\mathsf{T}_{i+3} - \mathsf{T}_{i+1}} \left\{ \frac{t - \mathsf{T}_{i+1}}{\mathsf{T}_{i+2} - \mathsf{T}_{i+1}} N_{i+1}^0(t) + \frac{\mathsf{T}_{i+3} - t}{\mathsf{T}_{i+3} - \mathsf{T}_{i+2}} N_{i+2}^0(t). \right\} \qquad (1.16) \end{aligned}$$

$$N_i^2(t) = \begin{cases} \dfrac{t - \mathsf{T}_i}{\mathsf{T}_{i+2} - \mathsf{T}_i} \cdot \dfrac{t - \mathsf{T}_i}{\mathsf{T}_{i+1} - \mathsf{T}_i} \ldots\ldots\ldots\ldots\ldots\ldots & \text{if } \mathsf{T}_i \le t < \mathsf{T}_{i+1}, \\ \dfrac{t - \mathsf{T}_i}{\mathsf{T}_{i+2} - \mathsf{T}_i} \cdot \dfrac{\mathsf{T}_{i+2} - t}{\mathsf{T}_{i+2} - \mathsf{T}_{i+1}} + \dfrac{\mathsf{T}_{i+3} - t}{\mathsf{T}_{i+3} - \mathsf{T}_{i+1}} \cdot \dfrac{t - \mathsf{T}_{i+1}}{\mathsf{T}_{i+2} - \mathsf{T}_{i+1}} & \text{if } \mathsf{T}_{i+1} \le t < \mathsf{T}_{i+2}, \\ \dfrac{\mathsf{T}_{i+3} - t}{\mathsf{T}_{i+3} - \mathsf{T}_{i+1}} \cdot \dfrac{\mathsf{T}_{i+3} - t}{\mathsf{T}_{i+3} - \mathsf{T}_{i+2}} \ldots\ldots\ldots\ldots\ldots & \text{if } \mathsf{T}_{i+2} \le t < \mathsf{T}_{i+3}, \\ 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \text{otherwise.} \end{cases} \qquad (1.17)$$

□

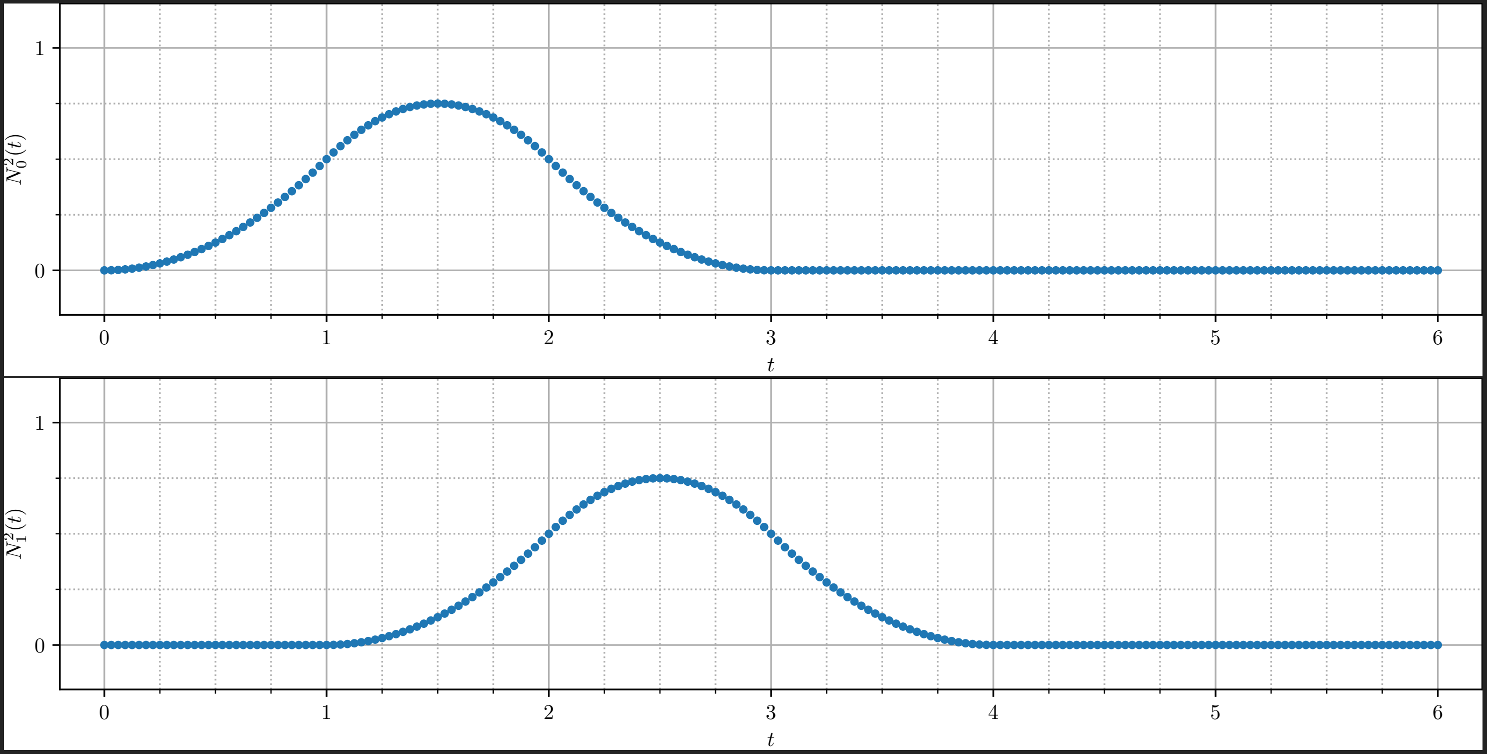

Figure 1.7: B-spline quadratic ($p = 2$) basis function. See `plot_bspline_basis_manual.py` on GitHub.

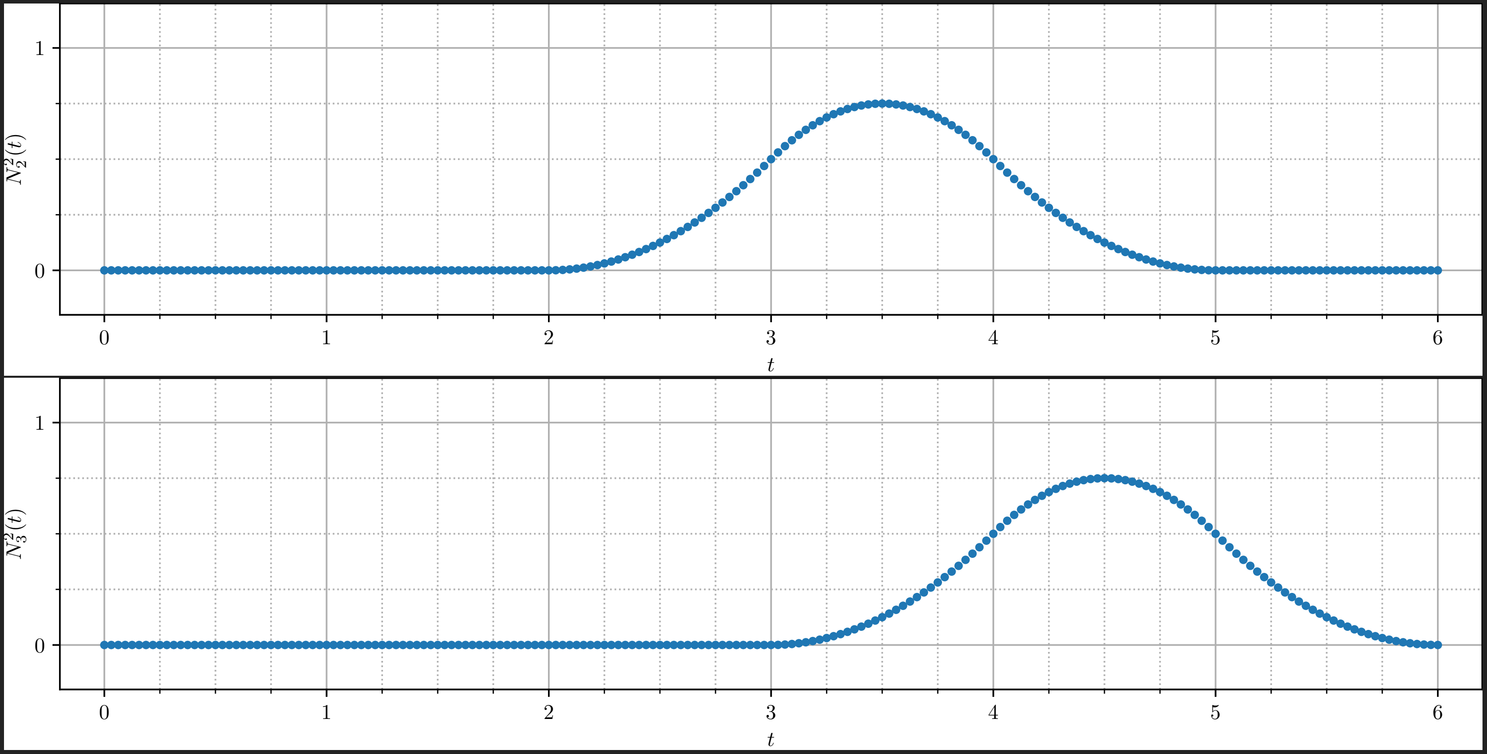

Figure 1.8: Continued from previous figure.

From 1.8: $N_i^p(t) \triangleq \dfrac{t - \mathsf{T}_i}{\mathsf{T}_{i+p} - \mathsf{T}_i} N_i^{p-1}(t) + \dfrac{\mathsf{T}_{i+p+1} - t}{\mathsf{T}_{i+p+1} - \mathsf{T}_{i+1}} N_{i+1}^{p-1}(t).$

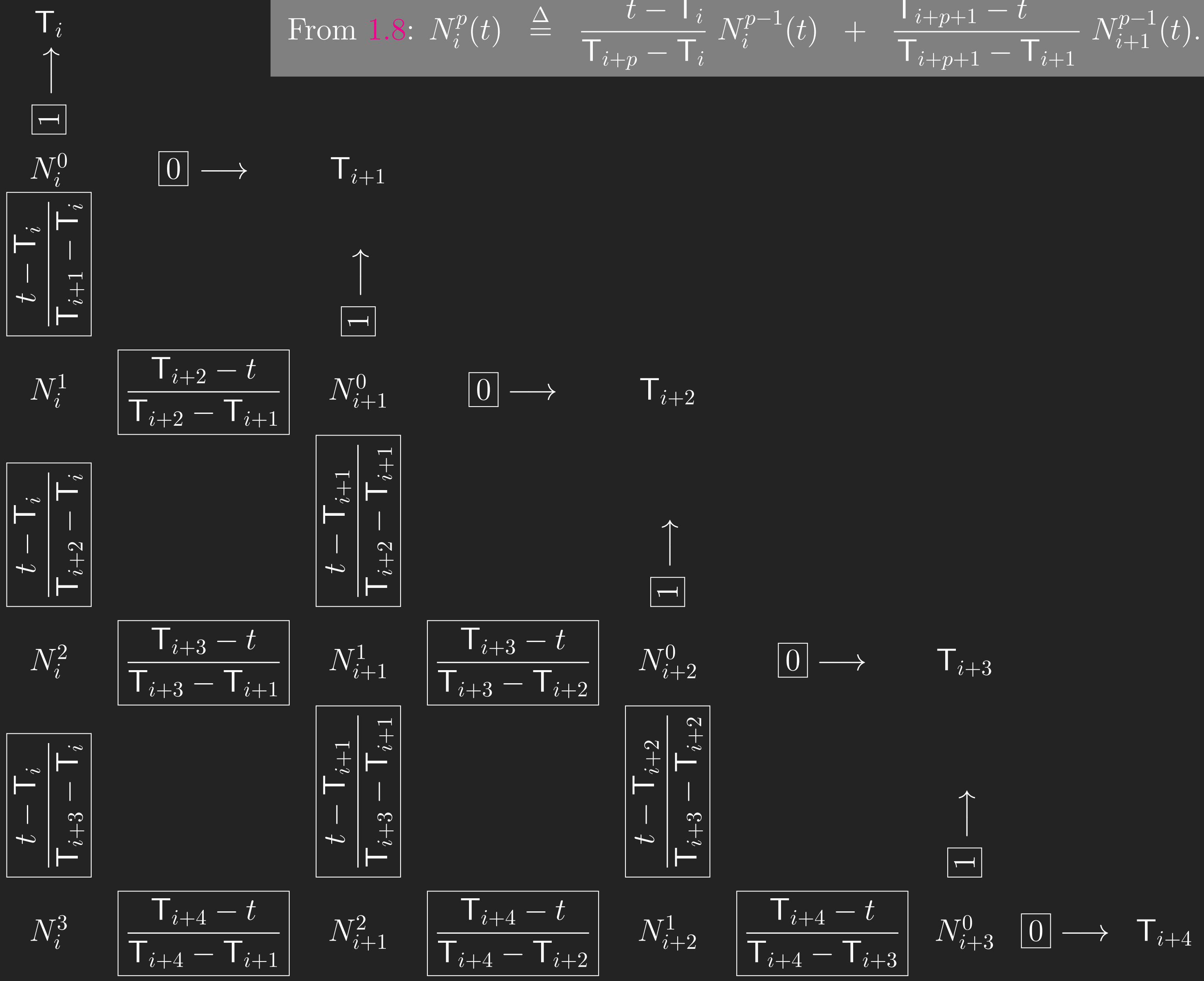

Figure 1.9: Cox-de Boor illustration up to cubic ($p = 3$).

Example 6.
The first B-spline basis functions, $N_0^p(t)$ for degrees $p \in [0, 1, 2, 3, 4]$, are plotted over $(p + 1)$ knot spans where there is local support. The knot vector is composed of five knots $\mathsf{\mathbf{T}} = \{\mathsf{T}_i\}_{i=0}^4 = \{\ 0,\ 1,\ 2,\ 3,\ 4,\ 5\ \}$. Note that each basis function has `(span = degree + 1)`, as shown in Fig. 1.10. □

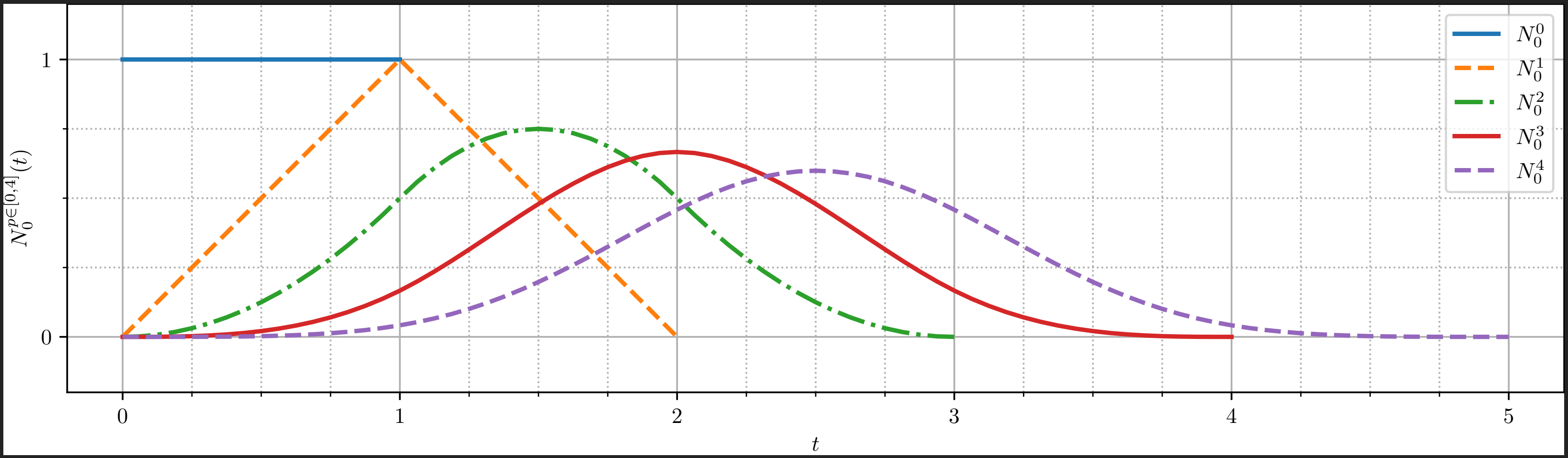

Figure 1.10: The first B-spline basis functions for degrees $p = 0$ to $p = 4$. See `plot_bspline_N0_p0_to_p4.py` on GitHub.

1.5 Non-Uniform Knot Vectors

The repetition of a knot value in the knot vector causes a knot span to go to zero, which is one way to cause knot vector to change from uniform to non-uninform.[3]

From this point forward, unless otherwise indicated, we focus on a special case of non-uniform knot vectors called **open knot vectors**,[4]

$$\mathsf{\mathbf{T}} = \{ \underbrace{\mathsf{T}_a, \ldots, \mathsf{T}_a}_{p+1}, \ \mathsf{T}_{p+1}, \ \ldots, \ \mathsf{T}_{\mathtt{I}-p-1}, \ \underbrace{\mathsf{T}_b, \ldots, \mathsf{T}_b}_{p+1} \}, \tag{1.18}$$

where the first knot value, $\mathsf{T}_a \overset{\text{set}}{=} \mathsf{T}_0$, and the last knot value, $\mathsf{T}_b \overset{\text{set}}{=} \mathsf{T}_{\mathtt{I}}$, are repeated $(p+1)$ times.

[3] The other way to cause a uniform knot vector to become non-uniform without repeated knot values is to have two or more knot spans with non-equal (and non-zero because repeated knot values are absent) knot interval distance.

[4] Open knot vectors are sometimes also called *clamped* knot vectors or *non-periodic* knot vectors.

- In Section 1.5.1, we introduce non-uniform knot vectors by reviewing cases where the first and last knots are repeated one or more times.
- In Section 1.5.2, we see how results in the preceding section can give rise to the Bézier basis functions as a special case of the B-spline basis functions.
- In Section 1.5.3, we examine repeated knots that are repeated in general throughout the knot vector (both at the knot vector endpoints as well as within the knot vector).
- In Section 1.5.4, we generalized the B-spline basis functions further, by allowing for non-uniform (and non-zero) knot spans within the knot vector.

1.5.1 Repeated Knot Values at Knot Vector Endpoints

Example 7.

The nine B-spline linear basis functions ($p = 1$) for the knot vector composed of 11 knots $\mathsf{T} = \{\mathsf{T}_i\}_{i=0}^{10} = \{\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 8\ \}$ produce eight elements (eight non-zero knot spans) as shown in Fig. 1.11. □

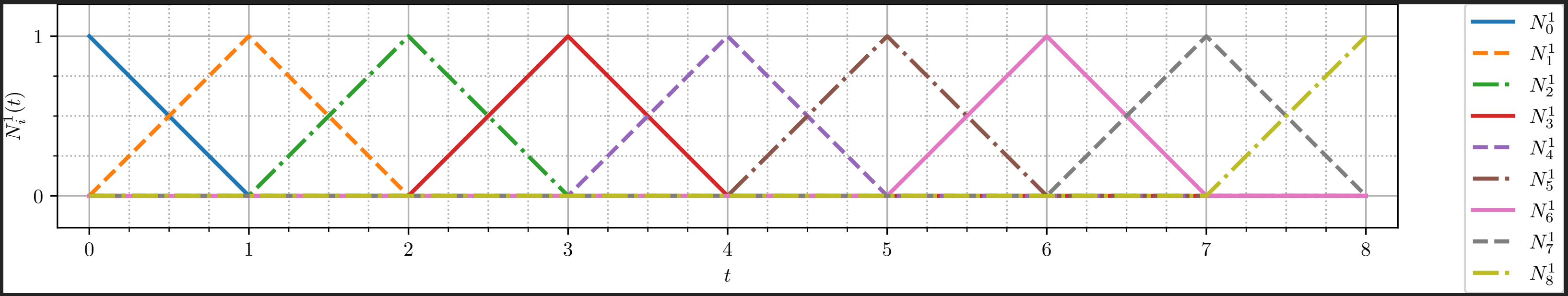

Figure 1.11: Nine B-spline linear basis functions. See `view_bspline.py` and `linear_expanded.json` on GitHub.

Example 8.

The nine B-spline quadratic basis functions ($p = 2$) for the knot vector composed of 12 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{11} = \{\ 0,\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 7,\ 7\ \}$ produce seven elements (seven non-zero knot spans) as shown in Fig. 1.12. □

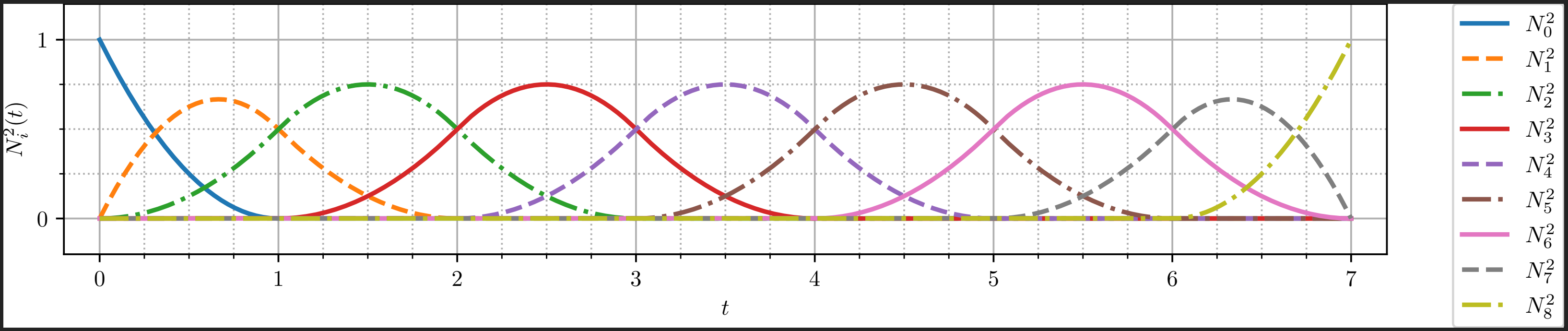

Figure 1.12: Nine B-spline quadratic basis functions. See `view_bspline.py` and `quadratic_expanded.json` on GitHub.

Example 9.
The bases for the periodic sections of the B-spline quadratic basis functions ($p = 2$) in the previous example (from element 1 to element 5), can be obtained from reformulation of the Bézier quadratic bases functions. Recall the Bézier quadratic curve, which used three basis functions to interpolate three controls, took the form:

$$\mathbb{C}^2(t) = B_0^2(t)\boldsymbol{P}_0 + B_1^2(t)\boldsymbol{P}_1 + B_2^2(t)\boldsymbol{P}_2, \tag{1.19}$$

$$= (1-t)^2\boldsymbol{P}_0 + 2(1-t)t\boldsymbol{P}_1 + t^2\boldsymbol{P}_2. \tag{1.20}$$

The periodic quadratic B-spline basis functions, shown in Fig. 1.12 and called the **periodic modified quadratic Bézier basis functions**, split the quadratic Bézier beginning and ending functions into two, and lets each half be weighted by center control point $\boldsymbol{P}_1$ as follows:

$$\hat{\mathbb{C}}^2(t) = \hat{N}_0^2(t)\boldsymbol{P}_0 + \hat{N}_1^2(t)\boldsymbol{P}_1 + \hat{N}_2^2(t)\boldsymbol{P}_2, \tag{1.21}$$

$$= \frac{1}{2}(1-t)^2\boldsymbol{P}_0 + \left(\frac{1}{2}(1-t)^2 + 2(1-t)t + \frac{1}{2}t^2\right)\boldsymbol{P}_1 + \frac{1}{2}t^2\boldsymbol{P}_2. \tag{1.22}$$

$$= \frac{1}{2}\ \underbrace{\langle\ \boldsymbol{P}_0 \quad \boldsymbol{P}_1 \quad \boldsymbol{P}_2\ \rangle}_{\substack{\text{control points}\\ \text{curve dependent}}}\ \underbrace{\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 1 \\ 1 & 0 & 0 \end{bmatrix}\begin{Bmatrix} t^2 \\ t \\ 1 \end{Bmatrix}}_{\text{curve independent}}. \tag{1.23}$$

Note the matrix is non-symmetric. For the Bézier, the matrix was symmetric. □

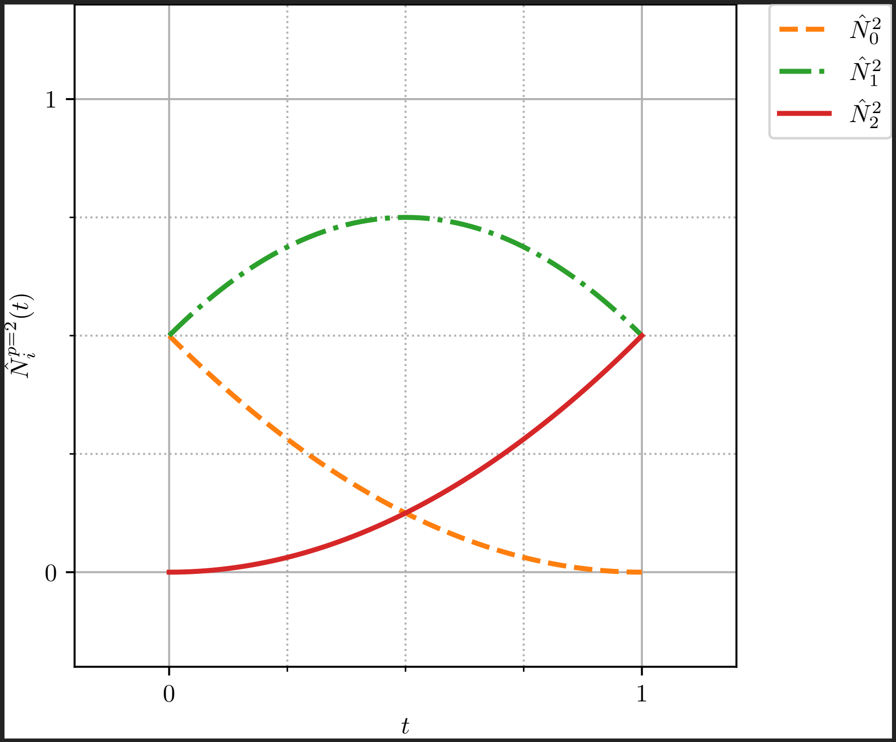

Figure 1.13: Periodic modified quadratic Bézier basis functions to recover periodic quadratic B-spline bases. See `plot_periodic_bspline_basis_p2.py` on GitHub.

Example 10.

The bases for the open (non-periodic) sections of the B-spline quadratic basis functions ($p = 2$) in the previous examples (element 0, and by symmetry, element 6), can be obtained from reformulation of the Bézier quadratic bases functions. Using the results from the previous example, we simply give back the one-half portion from the middle basis function to the first or the last basis function, to open the first or the last section of the spline. Recall the Bézier quadratic curve, which used three basis functions to interpolate three controls, took the form:

$$\begin{aligned} \mathbb{C}^2(t) &= B_0^2(t)\boldsymbol{P}_0 + B_1^2(t)\boldsymbol{P}_1 + B_2^2(t)\boldsymbol{P}_2, && (1.24)\\ &= (1-t)^2\boldsymbol{P}_0 + 2(1-t)t\boldsymbol{P}_1 + t^2\boldsymbol{P}_2. && (1.25) \end{aligned}$$

The first open (non-periodic) quadratic B-spline basis function, shown in Fig. 1.12, splits the quadratic Bézier ending function, $B_2^2(t)$ into two parts, and moves one of the half-parts to center control point $\boldsymbol{P}_1$ as follows:

$$\begin{aligned} \breve{\mathbb{C}}^2(t) &= \breve{N}_0^2(t)\boldsymbol{P}_0 + \breve{N}_1^2(t)\boldsymbol{P}_1 + \breve{N}_2^2(t)\boldsymbol{P}_2, && (1.26)\\ &= (1-t)^2\boldsymbol{P}_0 + \left(2(1-t)t + \frac{1}{2}t^2\right)\boldsymbol{P}_1 + \frac{1}{2}t^2\boldsymbol{P}_2. && (1.27) \end{aligned}$$

These will be referred to as the **left-open, right-periodic modified quadratic Bézier basis functions**. Note that the first basis, $\breve{N}_0^2$, will fully interpolate $\boldsymbol{P}_0$ at $t = 0$, where the other two bases go to zero. Control point $\boldsymbol{P}_1$ will be most influenced by $\breve{N}_1^2$ just after the mid-interval. Control point $\boldsymbol{P}_2$ will equally influenced by $\breve{N}_1^2$ and $\breve{N}_2^2$ at $t = 1$. □

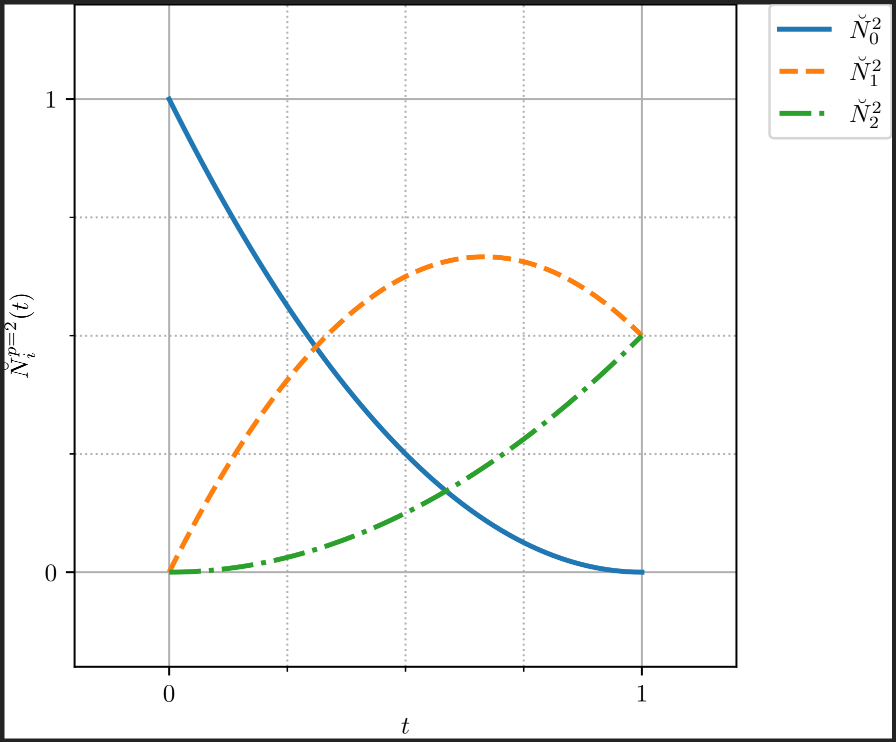

Figure 1.14: Left-open, right-periodic modified quadratic Bézier basis functions to recover first open (non-periodic) quadratic B-spline bases. See `plot_open_bspline_basis_p2.py` on GitHub.

Example 11.
The nine B-spline cubic basis functions ($p = 3$) for the knot vector composed of 13 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{12} = \{\ 0,\ 0,\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 6,\ 6,\ 6\ \}$ produce six elements (six non-zero knot spans) as shown in Fig. 1.15. □

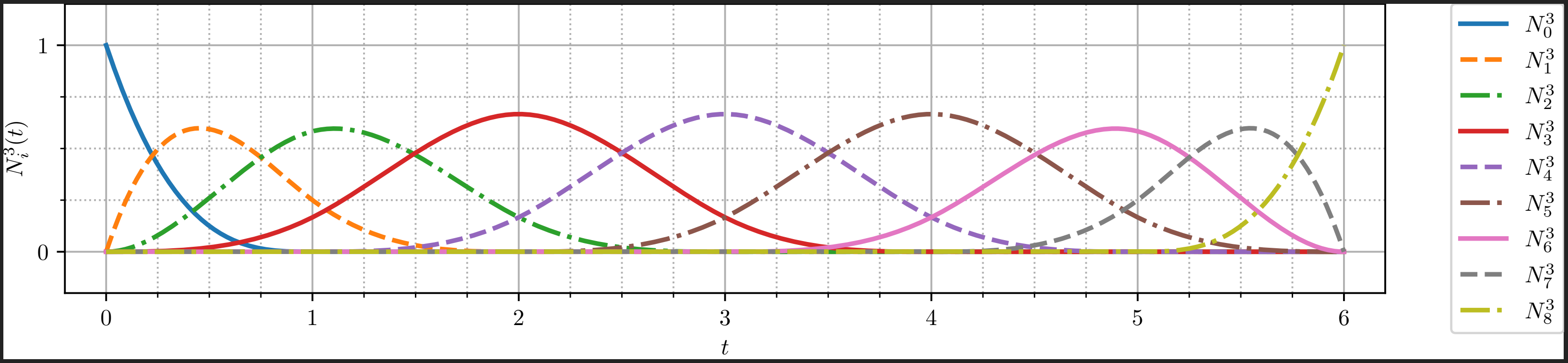

Figure 1.15: Nine B-spline cubic basis functions. See `view_bspline.py` and `cubic_expanded.json` on GitHub.

Example 12.

The nine B-spline quartic basis functions ($p = 4$) for the knot vector composed of 14 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{13} = \{\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 5,\ 5,\ 5,\ 5\ \}$ produce five elements (five non-zero knot spans) as shown in Fig. 1.16. □

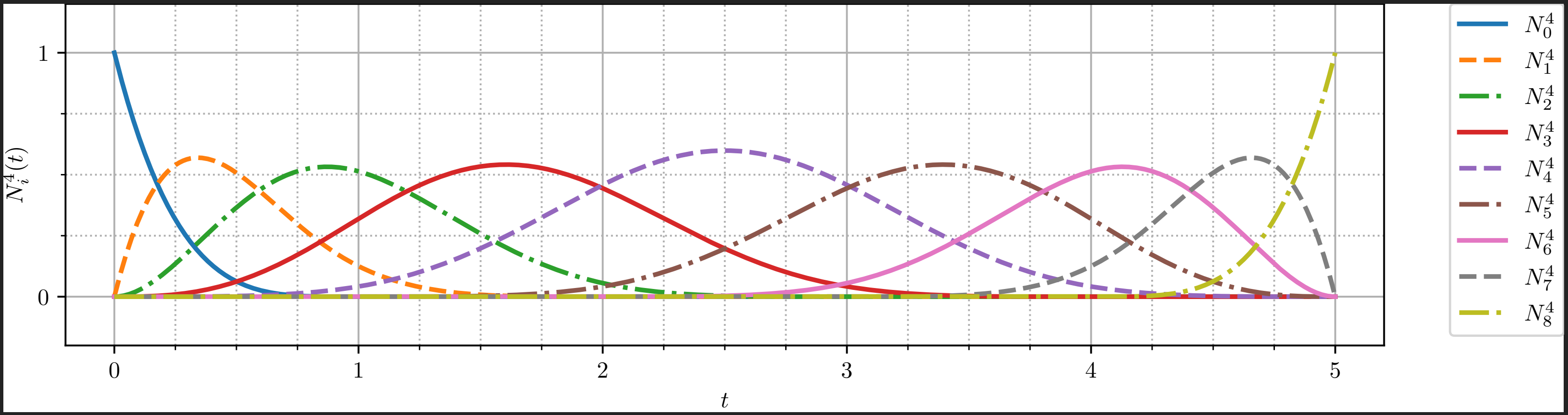

Figure 1.16: Nine B-spline quartic basis functions. See `view_bspline.py` and `quartic_expanded.json` on GitHub.

1.5.2 Recovery of Bézier Basis Functions

Bézier basis functions of degree p derive from B-spline basis functions that have knot vectors of the form

$$\mathbf{T} = \{\, \underbrace{0,\ \ldots,\ 0}_{p+1},\ \underbrace{1,\ \ldots,\ 1}_{p+1} \,\}. \tag{1.28}$$

Table 1.1 shows concrete examples of knot vectors for the linear, quadratic, cubic, and quartic cases. Figures 1.17 through 1.20 illustrate these cases graphically.

Table 1.1: Knot vectors for B-spline bases to recover Bézier bases.

degree	knot vector $\mathbf{T}$
$p = 1$	$\{\, 0,\ 0,\ 1,\ 1 \,\}$
$p = 2$	$\{\, 0,\ 0,\ 0,\ 1,\ 1,\ 1 \,\}$
$p = 3$	$\{\, 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1 \,\}$
$p = 4$	$\{\, 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1 \,\}$

Example 13.
Recovery of **Bézier linear** basis as a special case (Fig. 1.17).
The two Bézier linear ($p = 1$) basis functions $\{B_i^1\}_{i=0}^1$ are obtained as a special case of the B-spline linear basis functions $\{N_i^1\}_{i=0}^1$ when the knot vector $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^3 = \{\ 0,\ 0,\ 1,\ 1\ \}$.
□

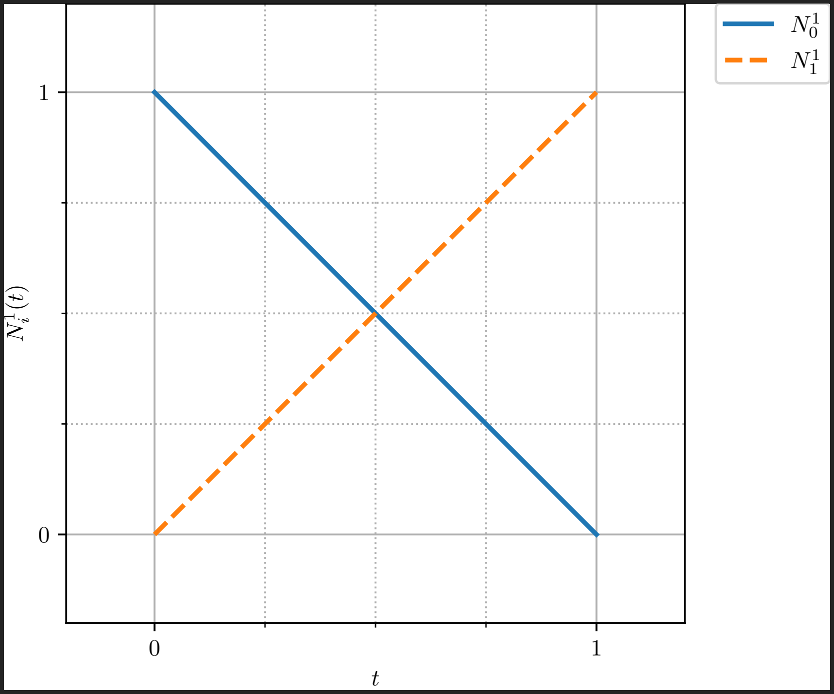

Figure 1.17: Recovery of Bézier linear basis functions from B-spline linear basis functions. See `view_bspline.py` and `recover_bezier_linear.json` on GitHub.

Example 14.
Recovery of **Bézier quadratic** basis as a special case (Fig. 1.18).
The three Bézier quadratic $(p = 2)$ basis functions $\{B_i^2\}_{i=0}^2$ are obtained as a special case of the B-spline quadratic basis functions $\{N_i^2\}_{i=0}^2$ when the knot vector $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^5 = \{\,0,\ 0,\ 0,\ 1,\ 1,\ 1\,\}$. $\square$

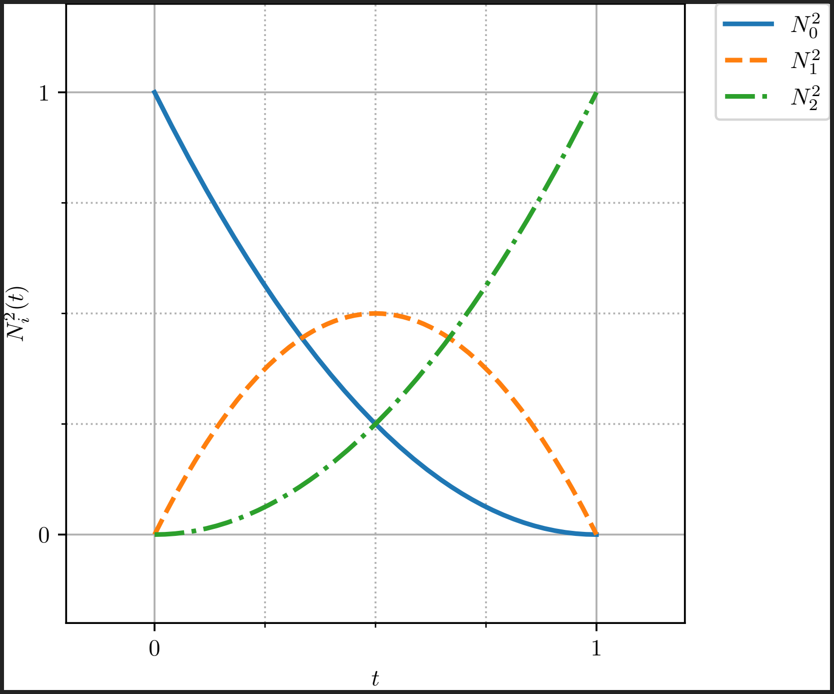

Figure 1.18: Recovery of Bézier quadratic basis functions from B-spline quadratic basis functions. See `view_bspline.py` and `recover_bezier_quadratic.json` on GitHub.

Example 15.
Recovery of **Bézier cubic** basis as a special case (Fig. 1.19).
The four Bézier cubic ($p = 3$) basis functions $\{B_i^3\}_{i=0}^3$ are obtained as a special case of the B-spline cubic basis functions $\{N_i^3\}_{i=0}^3$ when the knot vector $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^7 = \{ \, 0, \ 0, \ 0, \ 0, \ 1, \ 1, \ 1, \ 1 \, \}$.
□

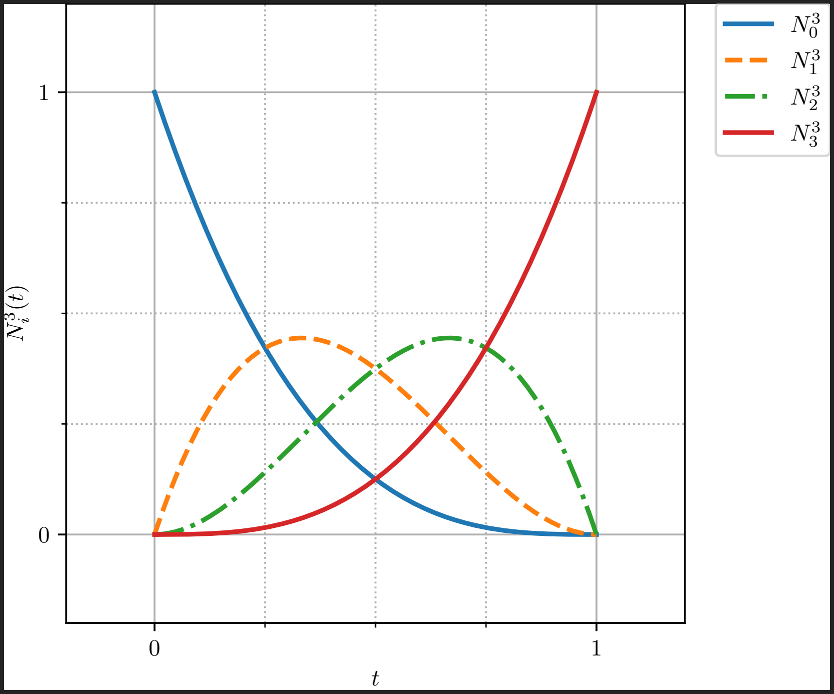

Figure 1.19: Recovery of Bézier cubic basis functions from B-spline cubic basis functions. See `view_bspline.py` and `recover_bezier_cubic.json` on GitHub.

Example 16.
Recovery of **Bézier quartic** basis as a special case (Fig. 1.20).
The five Bézier quartic $(p = 4)$ basis functions $\{B_i^4\}_{i=0}^4$ are obtained as a special case of the B-spline quartic basis functions $\{N_i^4\}_{i=0}^4$ when the knot vector $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^9 = \{ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1 \}$.
□

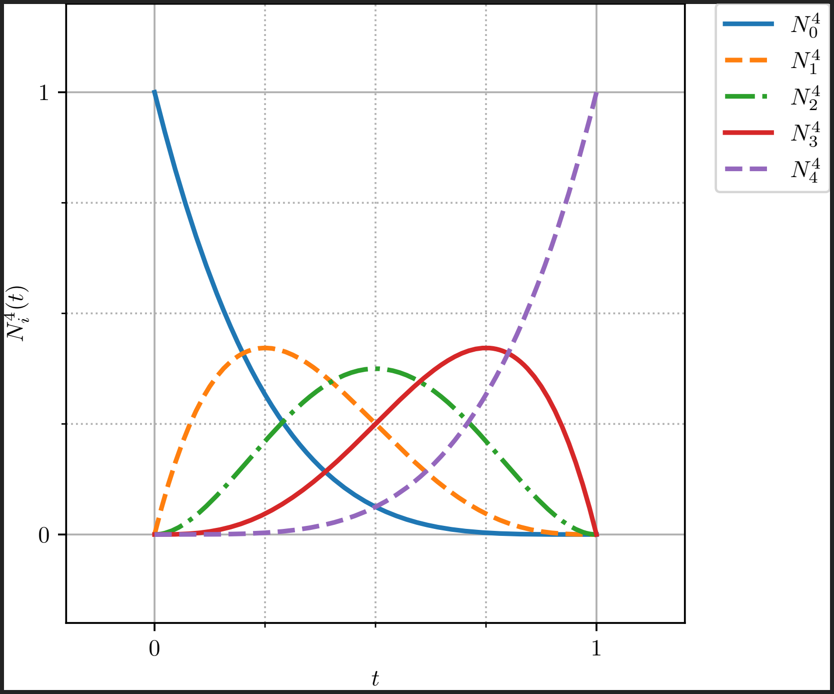

Figure 1.20: Recovery of Bézier quartic basis functions from B-spline quartic basis functions. See `view_bspline.py` and `recover_bezier_quartic.json` on GitHub.

Remark 1.5.6. Local Support
One important distinction: normalized basis functions of B-splines have local support; whereas, Bernstein polynomial basis of Béziers do *not* have local support.

For the B-spline basis, a single basis function is zero except for the spans over which it is defined as non-zero. Moving a knot, accomplished by changing its value, will modify only the bases that use that particular knot in a non-zero sense; all other bases remain unchanged. This is easy to conceptualize through study of Figure 1.1, were a single knot value increased or decreased.

In contrast, for the Bernstein polynomials, contributions from each basis function span the entire parameter domain. Bernstein polynomials provide global support, not local support.

Local support will be shown to be advantageous because a local modification to the curves, surfaces, and volumes created by B-splines will not alter the entire geometry; it only causes changes locally.

1.5.3 Repeated Knot Values In General

Example 17.
The eight B-spline quadratic basis functions ($p = 2$) for the knot vector composed of 11 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{10} = \{\ 0,\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 4,\ 5,\ 5,\ 5\ \}$ produce five elements (five non-zero knot spans) as shown in Fig. 1.21. These basese are used to contruct the B-spline curve in Fig. 2.3.
□

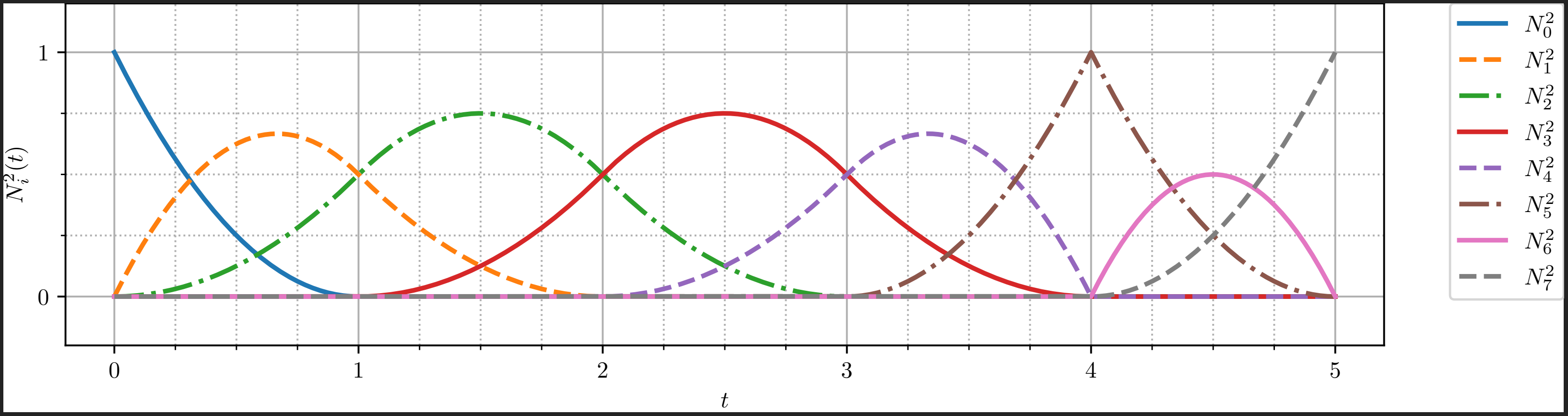

Figure 1.21: Reproduction of [Cottrell et al., 2009] Figure 2.5 (and [Piegl and Tiller, 1997] Figure 2.6). See `view_bspline.py` and `Cottrell_Fig2p5.json` on GitHub.

Example 18.

Reproduction of [Cottrell et al., 2009] Figure 2.6:

The 15 B-spline quartic basis functions ($p = 4$) for the knot vector composed of 20 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{19} = \{\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 2,\ 2,\ 3,\ 3,\ 3,\ 4,\ 4,\ 4,\ 4,\ 5,\ 5,\ 5,\ 5,\ 5\ \}$ produce five elements (five non-zero knot spans) as shown in Fig. 1.22. $\square$

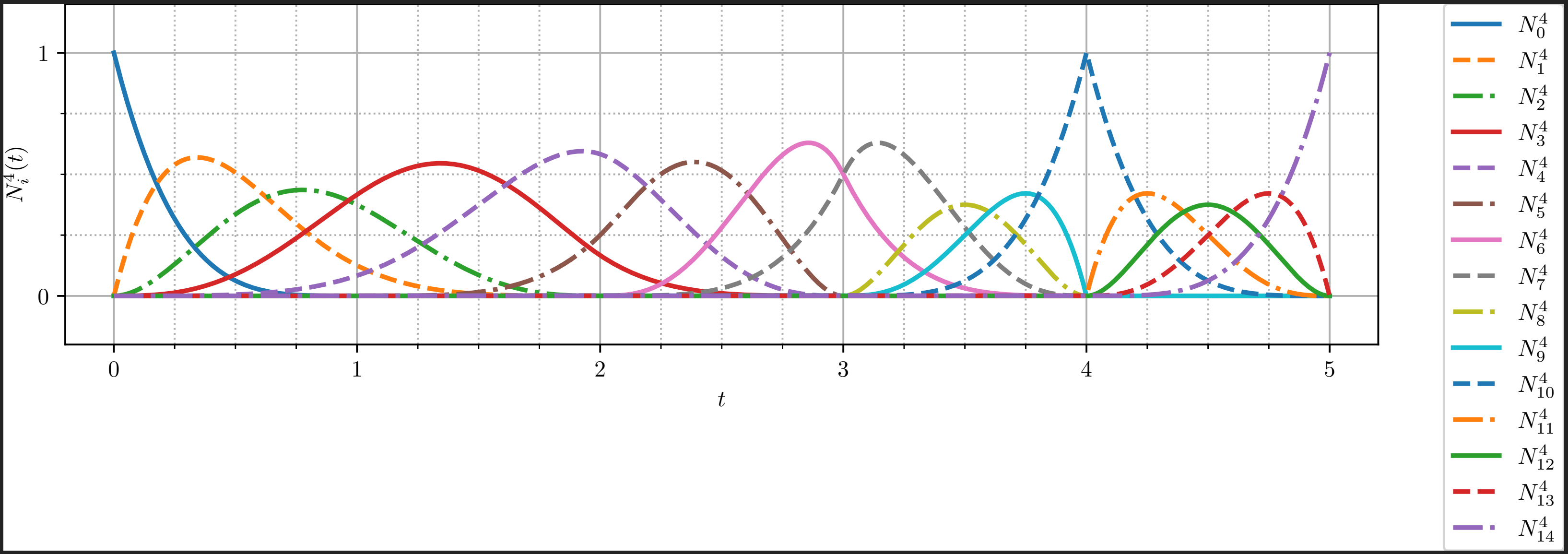

Figure 1.22: Reproduction of [Cottrell et al., 2009] Figure 2.6. See `view_bspline.py` and `Cottrell_Fig2p6.json` on GitHub.

1.5.4 Repeated Knot Values and Non-Zero, Non-Uniform Knot Spans

Example 19.
Reproduction of [Piegl and Tiller, 1997] Figure 2.12:
The seven B-spline cubic basis functions $(p = 3)$ for the knot vector composed of 11 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{10} = \{\ 0,\ 0,\ 0,\ 0,\ 1,\ 5,\ 6,\ 8,\ 8,\ 8,\ 8\ \}$ produce four elements (four non-zero knot spans) as shown in Fig. 1.23. $\square$

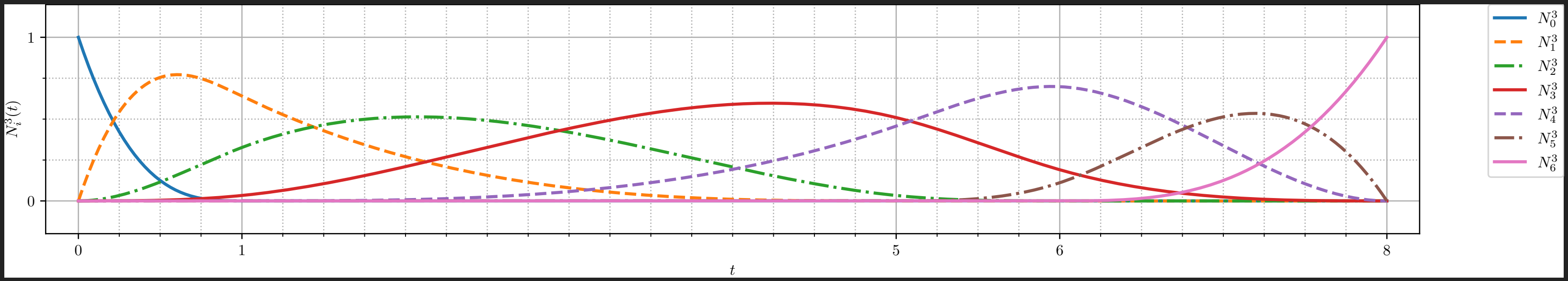

Figure 1.23: Reproduction of [Piegl and Tiller, 1997] Figure 2.12. See `view_bspline.py` and `Piegl_Fig2p12.json` on GitHub.

Chapter 2

B-Spline Curves

2.1 General Form

The general form of a degree p ($p \geq 0$) B-spline curve $\mathbb{C}^p(t)$ defined by $(n+1)$ control points $\{\boldsymbol{P}_i\}_{i=0}^{n}$ is given by

$$\mathbb{C}^p(t) \triangleq \sum_{i=0}^{n} N_i^p(t)\, \boldsymbol{P}_i, \qquad \text{for } t \in \mathbb{R} \subset [\,\mathsf{T}_0,\ \mathsf{T}_{\mathtt{I}}\,], \tag{2.1}$$

where $N_i^p(t)$ is a **B-spline basis function** of degree p, defined in (1.6) and (1.8), and t is a real number parameter bounded by the endpoints of the knot vector (see (1.2) and (1.3)).

2.2 Knot Dependence on Degree and Control Points

An open B-spline basis function of degree p
with $(n+1)$ control points will require
$(\mathtt{I})$ knot spans and thus $(\mathtt{I}+1)$ knots, where
$\mathtt{I} = p + n + 1$.

Equivalently,

$$\underbrace{(\ \mathtt{I}+1\ )}_{\#\text{ knots}} = \underbrace{(\ p+1\ )}_{\text{degree } + 1} + \underbrace{(\ n+1\ )}_{\#\text{ control points}} \tag{2.2}$$

Thus,

(# knots) = (degree + 1) + (# control points)
(# knots) = (order) + (# control points)

Table 2.1: Requirements for number of knot spans, given a B-spline of degree p, up to cubic ($p = 3$), and number of control points $(n+1)$.

degree	**control points**	$(n+1)$	**knot spans** m	**knot vector**	**parameter span**
$p=0$	$\boldsymbol{P}_0$	1	1	$\{\, \mathsf{T}_0, \mathsf{T}_1 \,\}$	$t \in [\mathsf{T}_0, \mathsf{T}_1)$
$p=1$	$\boldsymbol{P}_0, \boldsymbol{P}_1$	2	3	$\{\, \mathsf{T}_0, \mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3 \,\}$	$t \in [\mathsf{T}_0, \mathsf{T}_3)$
$p=2$	$\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2$	3	5	$\{\, \mathsf{T}_0, \mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4, \mathsf{T}_5 \,\}$	$t \in [\mathsf{T}_0, \mathsf{T}_5)$
$p=3$	$\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3$	4	7	$\{\, \mathsf{T}_0, \mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4, \mathsf{T}_5, \mathsf{T}_6, \mathsf{T}_7 \,\}$	$t \in [\mathsf{T}_0, \mathsf{T}_7)$

2.3 Verifications

Following are several examples that have been used as verification of the implementation on GitHub. Knots are evaluated and shown along the B-spline curve. In the case of repeated knots, only the first knot index is indicated.

Example 20.

A cubic $(p = 3)$ Bézier curve (Fig. 2.1) is a special case of a cubic B-spline curve (see basis functions and knot vector 1.19) with knot vector composed of eight knots $\mathsf{T} = \{\mathsf{T}_i\}_{i=0}^{7} = \{\ 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1\ \}$ (cyan circles), a single element (one non-zero knot span), and four control points $\{\boldsymbol{P}_i\}_{i=0}^{3} = \{\ (0,0),\ (0.6, 1.6),\ (2.1, 1.9),\ (3,0)\ \}$ (red circles). $\square$

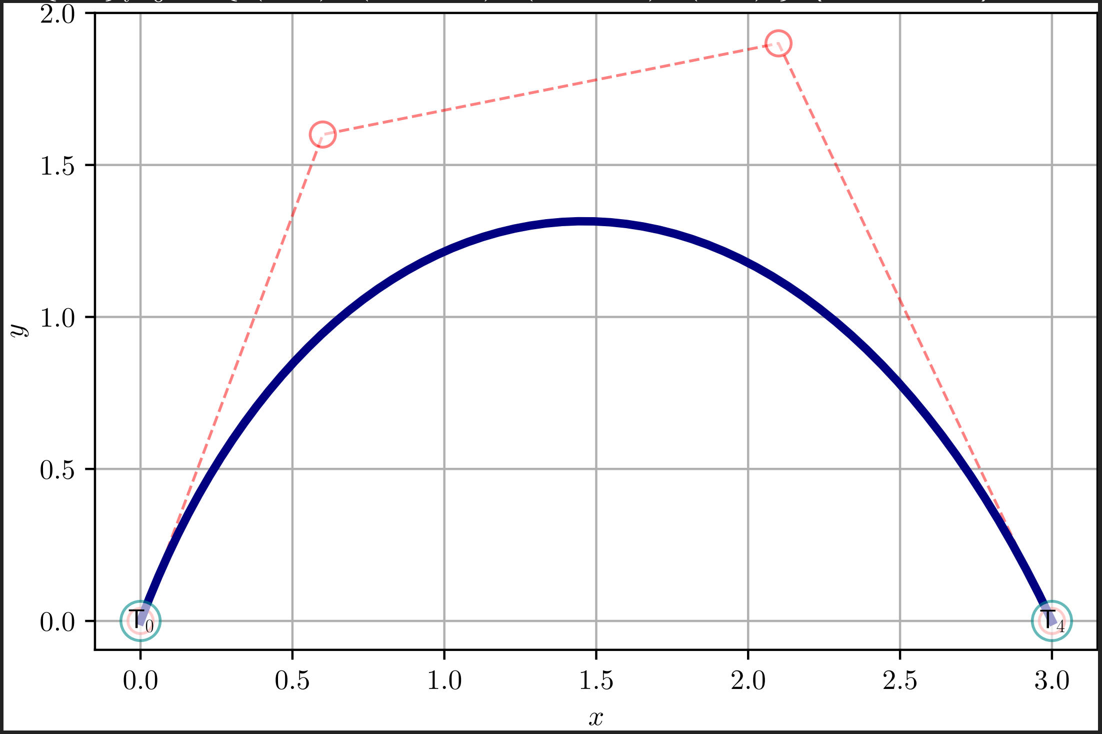

Figure 2.1: Reproduction of [Piegl and Tiller, 1997] Figure 3.1. See `view_bspline.py` and `Piegl_Fig3p1.json` on GitHub.

Example 21.
A cubic $(p = 3)$ B-spline curve (Fig. 2.2) with knot vector composed of 11 knots $\mathbf{T} = \{\mathsf{T}_i\}_{i=0}^{10} = \{\ 0,\ 0,\ 0,\ 0,\ 0.25,\ 0.50,\ 0.75,\ 1,\ 1,\ 1,\ 1\ \}$ (cyan circles), four elements (four non-zero knot spans), and seven control points $\{\boldsymbol{P}_i\}_{i=0}^{6} = \{\ (-14, 0),\ (0, 0),\ (0, 13),\ (15, 13),\ (20, -1.5),\ (9, -10),\ (0, -5)\ \}$ (red circles). $\square$

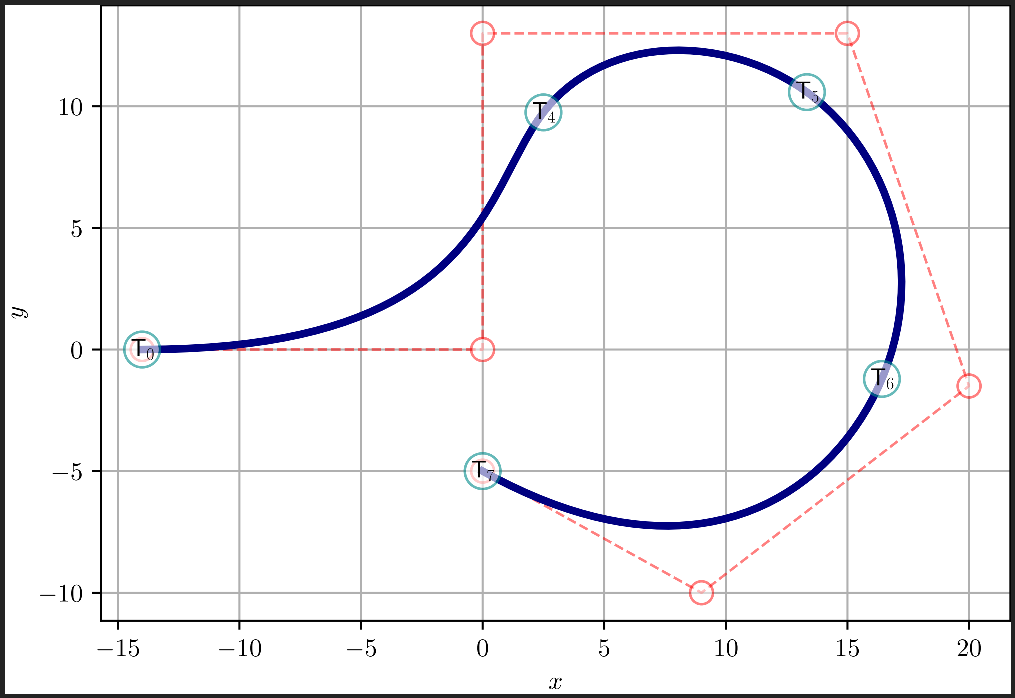

Figure 2.2: Reproduction of [Piegl and Tiller, 1997] Figure 3.2. See `view_bspline.py` and `Piegl_Fig3p2.json` on GitHub.

Example 22.
A quadratic ($p = 2$) B-spline curve (Fig. 2.3) with knot vector composed of 11 knots $\mathsf{T} = \{\mathsf{T}_i\}_{i=0}^{10} = \{\ 0,\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 4,\ 5,\ 5,\ 5\ \}$ (cyan circles), five elements (five non-zero knot spans), and eight control points $\{\boldsymbol{P}_i\}_{i=0}^{7} = \{\ (0,1),\ (1,0),\ (2,0),\ (2,2),\ (4,2),\ (5,4),\ (2,5),\ (1,3)\ \}$ (red circles). $\square$

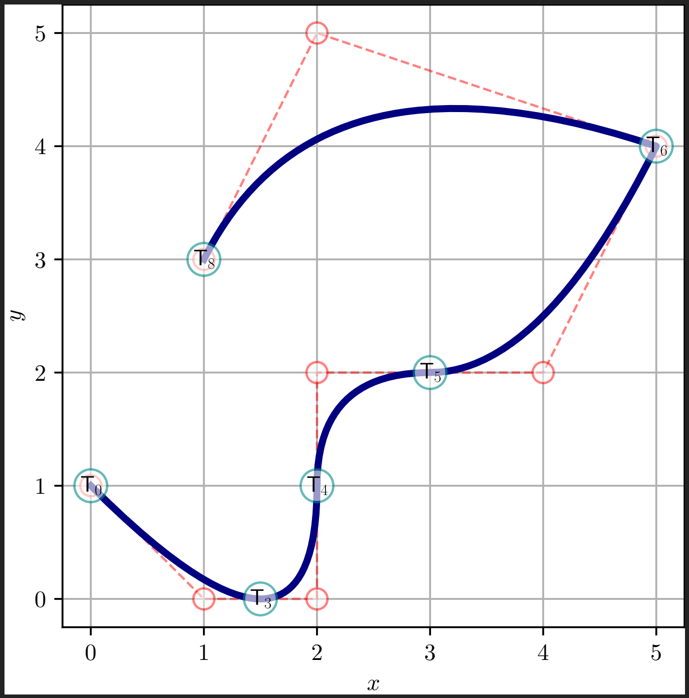

Figure 2.3: Reproduction of [Cottrell et al., 2009] Figure 2.20. See `view_bspline.py` and `Cottrell_Fig2p20.json` on GitHub.

Example 23.

A quartic ($p = 4$) B-spline curve (Fig. 2.4) with knot vector composed of 14 knots $\mathsf{\mathbf{T}} = \{\mathsf{T}_i\}_{i=0}^{13} = \{\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1,\ 1,\ 1,\ 1,\ 1\ \}$ (cyan circles), five elements (five non-zero knot spans), and nine control points $\{\boldsymbol{P}_i\}_{i=0}^{8} = \{\ (5,10),\ (15,25),\ (30,30),\ (45,5),\ (55,5),\ (70,40),\ (60,60),\ (35,60),\ (20,40)\ \}$ (red circles). $\square$

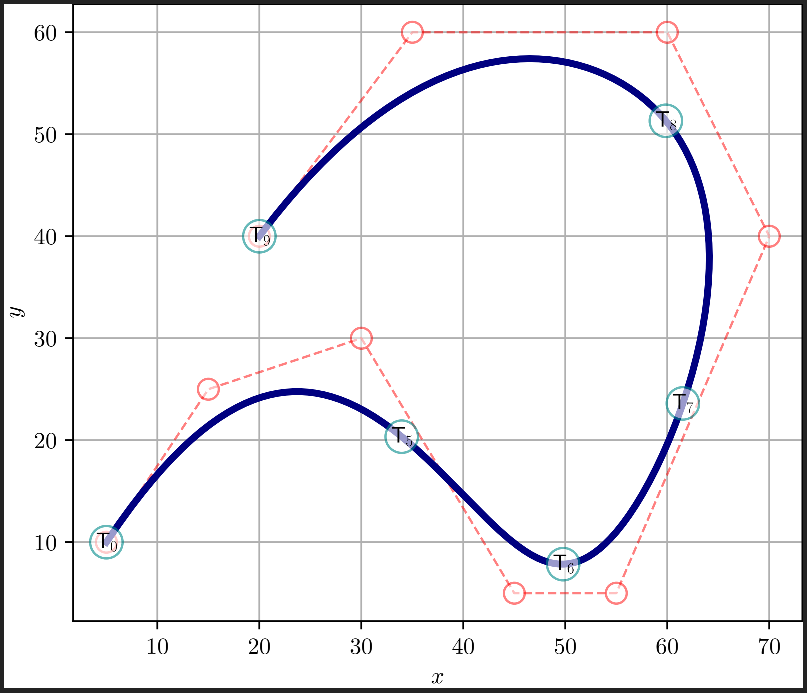

Figure 2.4: Reproduction of [Bingol and Krishnamurthy, 2019] example. See `view_bspline.py` and `Bingol_2D_curve.json` on GitHub.

2.4 Additional Examples

Example 24.
A quadratic ($p = 2$) B-spline curve (Fig. 2.5) with knot vector composed of 12 knots $\mathsf{T} = \{\mathsf{T}_i\}_{i=0}^{11} = \{\ 0,\ 0,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 7,\ 7\ \}$ (cyan circles), seven elements (seven non-zero knot spans), and nine control points $\{\boldsymbol{P}_i\}_{i=0}^{8} = \{\ (1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1), (1,0)\ \}$ (red circles). Note the asymmetry about the $x = 0$ axis, caused by the open knot vector's repeated knots, beginning and end. $\square$

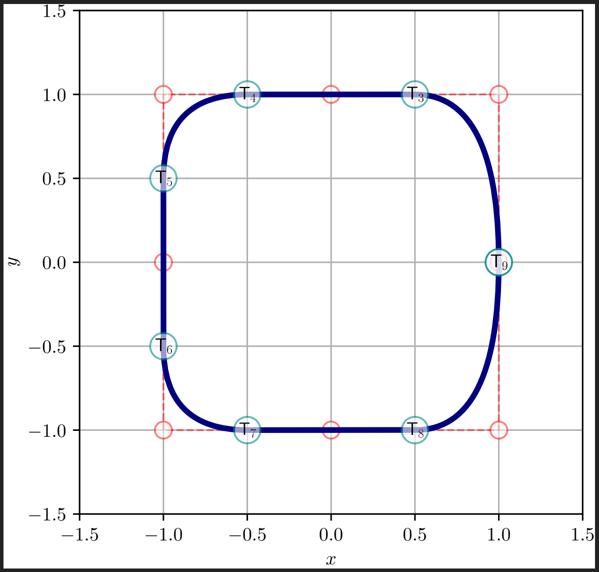

Figure 2.5: See `view_bspline.py` and `circle_curve_9_points.json` on GitHub.

Example 25.
A periodic modified quadratic ($p = 2$) Bézier curve (Fig. 2.6) with eight elements, labeled ⓪ to ⑦, and eight control points $\{\boldsymbol{P}_i\}_{i=0}^{7} = \{\, (1, 0),\ (1, 1),\ (0, 1),\ (-1, 1),\ (-1, 0),\ (-1, -1),\ (0, -1),\ (1, -1)\, \}$. $\square$

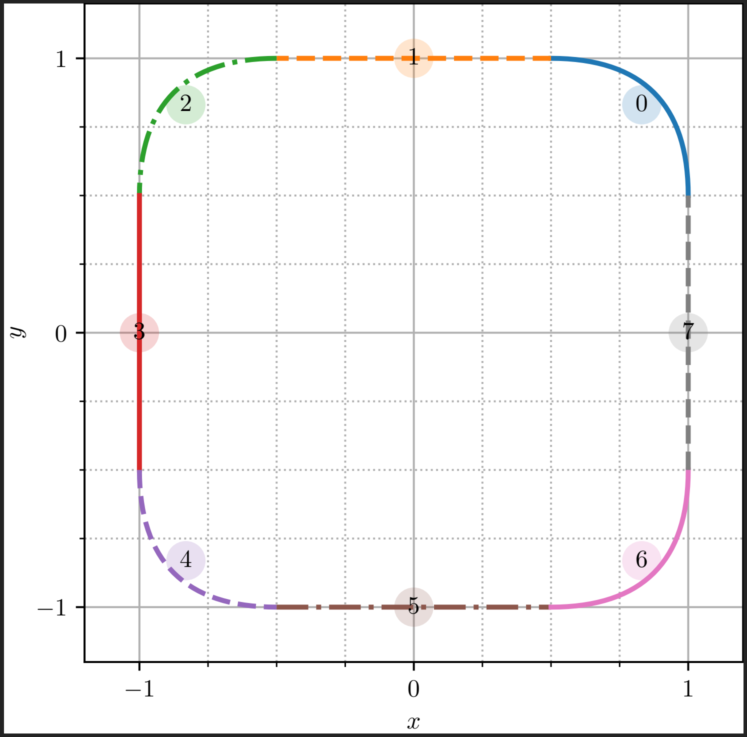

Figure 2.6: See `plot_modified_bezier.py` and `circle-points.csv` on GitHub.

Example 26.
A periodic modified linear ($p = 1$) Bézier curve (Fig. 2.7) with eight elements, labeled $\textcircled{0}$ to $\textcircled{7}$, and eight control points $\{\boldsymbol{P}_i\}_{i=0}^{7} = \{\,(1,0),\,(1,1),\,(0,1),\,(-1,1),\,(-1,0),\,(-1,-1),\,(0,-1),\,(1,-1)\,\}$. $\square$

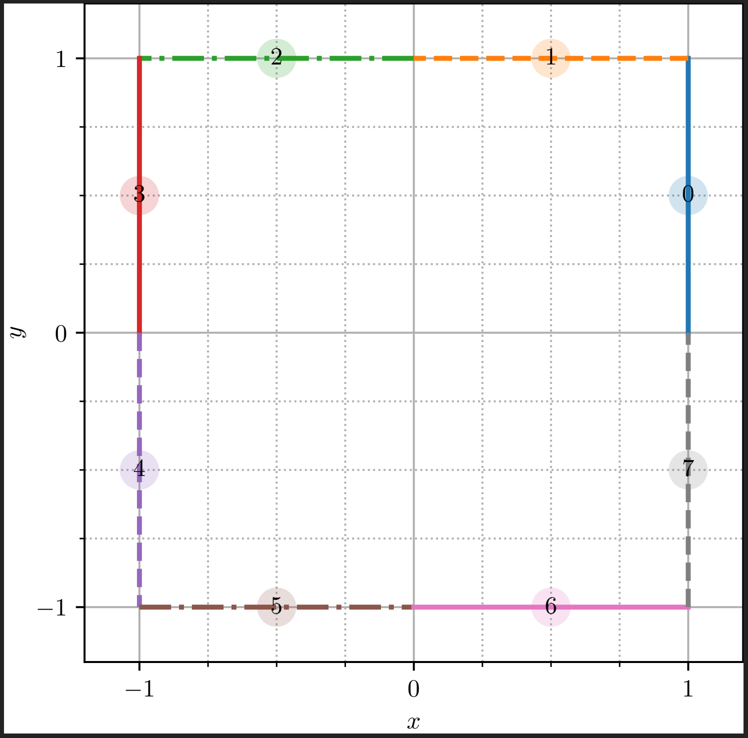

Figure 2.7: See `plot_modified_bezier.py` and `circle-points.csv` on GitHub.

Chapter 3

Curves from Sample Points

In this chapter, we demonstrate how to create B-Spline curve with *a priori unknown* control points from a set of *known* data points, sampled from an arbitrary curve.

- In prior chapters, we used a set of $(n{+}1)$ *known* control points $\{\boldsymbol{P}_i\}_{i=0}^n = \{\ x_i,\ y_i,\ z_i\ \}_{i=0}^n$ to generate a curve $\mathbb{C}(t)$ composed of points in space $\{\ x(t),\ (y)t,\ z(t)\ \}$, parameterized by pseudo-time $t \in [\ \mathsf{T}_0,\ \mathsf{T}_{\mathtt{I}}\ ]$.

- Now we do the inverse problem: We use a set of $(s{+}1)$ *known* data points $\{\ \boldsymbol{\Pi}_k\ \}_{k=0}^s = \{\ \alpha_k,\ \beta_k,\ \gamma_k\ \}_{k=0}^s$ sampled at (unknown) time $\tau_k \in [\ \tau_0,\ \tau_s\ ]$ that lie on or near a curve $\mathbb{C}(\tau)$ generated from $(n+1)$ *unknown* control points $\{\tilde{\boldsymbol{P}}\}_{i=0}^n = \{\ \tilde{x}_i,\ \tilde{y}_i,\ \tilde{z}_i\ \}_{i=0}^n$. The generated curve will fit to the given sampled points, up to some error tolerance.

3.1 Development of a Curve Fit Methodology

[Piegl and Tiller, 1997][1] described two methodologies, **interpolation** and **approximation**, to solve the inverse problem.

- **Interpolation** satisfies the sample data precisely, and leads to equation solving of a square matrix of dimension $(n+1) \times (n+1)$.
- **Approximation** does not necessarily satisfy the sample data precisely, and leads to least-squares equation solving from an over-constrained matrix of dimension $(s+1) \times (n+1)$, where $s > n$, described by [Piegl and Tiller, 1997] and [Eberly, 2020].

In the present work, we develop a curve fit methodology based on *approximation*, and present *interpolation* as a special case (*i.e.*, when $s = n$). We denote $(s+1)$ to be the number of sample points. We denote $(n+1)$ to be the number of control points.

[1] At 361.

3.1.1 Sample Points and Control Points Relationship

Let the set of $(s+1)$ sample points $\{\ \mathbf{\Pi}_k\ \}_{k=0}^{s}$ have coordinates $\{\ \alpha_k,\ \beta_k,\ \gamma_k\ \}_{k=0}^{s}$ measured at psuedo-time $\tau_k \in [\ \tau_0,\ \ \tau_s\ ]$. Imagine each sample point satisfies the B-spline curve equation (5.7) of degree p with $(n+1)$ control points such that

$$\{\ \alpha(\tau),\ \beta(\tau),\ \gamma(\tau)\ \} = \mathbb{C}^p(\tau) = \sum_{i=0}^{n} N_i^p(\tau)\ \tilde{\boldsymbol{P}}_i. \qquad \text{for } \tau \in \mathbb{R} \subset [\ \tau_0,\ \ \tau_s\ ]. \tag{3.1}$$

At discrete sample times, the foregoing equation can be expanded as[2]

$$\underbrace{\begin{bmatrix} \alpha_0 & \beta_0 & \gamma_0 \\ \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \vdots & \vdots & \vdots \\ \alpha_s & \beta_s & \gamma_s \end{bmatrix}}_{(s+1)\ \times\ \texttt{nsd}} = \underbrace{\begin{bmatrix} N_0(\tau_0) & N_1(\tau_0) & \cdots & N_n(\tau_0) \\ N_0(\tau_1) & N_1(\tau_1) & \cdots & N_n(\tau_1) \\ N_0(\tau_2) & N_1(\tau_2) & \cdots & N_n(\tau_2) \\ \vdots & \vdots & \ddots & \vdots \\ N_0(\tau_s) & N_1(\tau_s) & \cdots & N_n(\tau_s) \end{bmatrix}}_{(s+1)\ \times\ (n\times 1)} \underbrace{\begin{bmatrix} \tilde{x}_0 & \tilde{y}_0 & \tilde{z}_0 \\ \tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\ \vdots & \vdots & \vdots \\ \tilde{x}_n & \tilde{y}_n & \tilde{z}_n \end{bmatrix}}_{(n+1)\ \times\ \texttt{nsd}}. \tag{3.2}$$

The number of knots, $(\mathtt{I}+1)$, remains a function of the basis function degree and the number of control points through Eq. (2.2). Open knot vectors are used, so the knot vector maintains the form in Eq. (1.18). For convenience, we set $\tau_0 = 0$ and $\tau_s = 1$. Thus $\mathsf{T}_0 = \tau_0 = 0$ and $\mathsf{T}_\mathtt{I} = \tau_s = 1$.

[2]For brevity, the degree p of the basis functions is omitted. Thus, $N_i^p(t)$ is simply $N_i(t)$, where p is a given input.

The number of space dimensions, indicated as `nsd`, is typically two or three for 2D or 3D, respectively. We refer to the $(s+1) \times (n+1)$ matrix as the **sample basis matrix $\mathbf{N}$**, since it represents evaluation of the B-spline normalized basis functions at sample times τ_k.

Taking each space dimension in isolation, the foregoing equation for x-axis data can be written as,

$$\underbrace{\begin{Bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_s \end{Bmatrix}}_{\dim(\boldsymbol{\alpha})=(s+1)\times 1} = \underbrace{\begin{bmatrix} N_0(\tau_0) & N_1(\tau_0) & \cdots & N_n(\tau_0) \\ N_0(\tau_1) & N_1(\tau_1) & \cdots & N_n(\tau_1) \\ N_0(\tau_2) & N_1(\tau_2) & \cdots & N_n(\tau_2) \\ \vdots & \vdots & \ddots & \vdots \\ N_0(\tau_s) & N_1(\tau_s) & \cdots & N_n(\tau_s) \end{bmatrix}}_{\dim(\mathbf{N})=(s+1)\times(n\times 1)} \underbrace{\begin{Bmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \vdots \\ \tilde{x}_n \end{Bmatrix}}_{\dim(\tilde{\boldsymbol{x}})=(n+1)\times 1}, \tag{3.3}$$

or simply,

$$\boldsymbol{\alpha} = \mathbf{N}\, \tilde{\boldsymbol{x}}. \tag{3.4}$$

Respectively, for y-axis and z-axis data, these equations are

$$\boldsymbol{\beta} = \mathbf{N}\, \tilde{\boldsymbol{y}}, \quad \text{and} \quad \boldsymbol{\gamma} = \mathbf{N}\, \tilde{\boldsymbol{z}}. \tag{3.5}$$

The $\mathbf{N}$ matrix depends on the pseudo-time parameter data $\boldsymbol{\tau} = \{\ \tau_k\ \}_{k=0}^{s}$.

How do we construct the interior sample times? Several strategies exist. The most simple strategy is to assume **equally spaced** time parameter intervals for each of the $(s+1)$ sample points, thus

$$\boldsymbol{\tau} = \{\, 0,\ \frac{1}{s},\ \frac{2}{s},\ \frac{3}{s},\ \ldots,\ \frac{s}{s}\,\} = \frac{1}{s}\{\, 0,\ 1,\ 2,\ 3,\ \ldots,\ s\,\}. \tag{3.6}$$

This strategy is not recommended for the interpolation case, as explained below.

3.1.2 The Interpolation Special Case $(s = n)$

For the case where the number of sample points equals the number of control points, *i.e.*, $(s+1) = (n+1)$, [Piegl and Tiller, 1997][3] do not recommend the equally spaced time parameter method (3.6) because it "can produce erratic shapes (such as loops) when the [sample point] data is unevenly spaced." They recommend two alternative methods. The first alternative, the **chord length method**, creates the following sample times from successive sample point distances:

$$\left.\begin{aligned} \tau_0 &= 0, \\ \tau_k &= \tau_{k-1} + \frac{|\,\boldsymbol{\Pi}_k - \boldsymbol{\Pi}_{k-1}\,|}{C},\ \text{for } k = 1, \ldots, (n-1), \\ \tau_n &= 1, \end{aligned}\right\} \tag{3.7}$$

[3] At 364.

where the **total chord length** C as is defined as

$$C \triangleq \sum_{k=1}^{n} |\, \mathbf{\Pi}_k - \mathbf{\Pi}_{k-1} \,|. \tag{3.8}$$

The physical interpretation of this method is constant velocity: That is, the pseudo-time interval scales directly with the distance between sample points. This can be seen readily by rearranging (3.7),

$$C = \frac{|\, \mathbf{\Pi}_k - \mathbf{\Pi}_{k-1} \,|}{\tau_k - \tau_{k-1}} \implies \frac{\text{change in position}}{\text{change in time}} = \text{velocity}. \tag{3.9}$$

The value of C is the total chord length, which is constant for a given set of sample points.

The second alternative method, the **centripetal method**, modifies (3.7) and (3.8) with a square root,

$$\left.\begin{aligned} &\tau_0 = 0, \\ &\tau_k = \tau_{k-1} + \frac{\sqrt{|\, \mathbf{\Pi}_k - \mathbf{\Pi}_{k-1} \,|}}{D}, \ \text{for } k = 1, \ldots, (n-1), \\ &\tau_n = 1, \end{aligned}\right\} \tag{3.10}$$

where

$$D \triangleq \sum_{k=1}^{n} \sqrt{|\, \mathbf{\Pi}_k - \mathbf{\Pi}_{k-1} \,|}. \tag{3.11}$$

The square root amplifies distances per unit characteristic length when the sample interval distance gets small (approaches zero), making it suited for sample data with "very sharp turns."[Piegl and Tiller, 1997][4]

Next we turn our attention to the knot vectors. The most elementary method is to use equally spaced knots,

$$\left.\begin{aligned} \mathsf{T}_0 &= \cdots = \mathsf{T}_p = 0, \\ \mathsf{T}_{p+j} &= \frac{j}{n+1-p}, \quad \text{for } j = 1, \ldots, (n-p), \\ \mathsf{T}_{\mathtt{I}-p} &= \cdots = \mathsf{T}_{\mathtt{I}} = 1. \end{aligned}\right\} \tag{3.12}$$

The following example illustrates how the equally spaced space knots strategy in (3.12) leads to recovery of the knot vectors used in Figures. 1.11 through 1.16.

Example 27.

Equally spaced knots. Revisiting the examples shown in as shown in Figures. 1.11 through 1.16, we demonstrate how the foregoing equations would produce equally spaced knot vectors. Here number of sample points $(s+1)$ is set equal to the number of control points $(n+1)$, which is nine for all cases (thus $s = n = 8$). Equation (2.2) is used to determine the number of knots.

[4]At 365.

Fig. #	degree p	# knots $(\mathtt{I}+1)$	knots $\mathsf{T}_0, \ldots, \mathsf{T}_\mathtt{I}$
1.11	1	11	$\mathsf{T}_0 = \mathsf{T}_1 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = 1,$ $\mathsf{T}_{1+j} = j/8$ for $j = 1, \ldots, 7$
1.12	2	12	$\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}_2 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = \mathsf{T}_{11} = 1,$ $\mathsf{T}_{2+j} = j/7$ for $j = 1, \ldots, 6$
1.15	3	13	$\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}_2 = \mathsf{T}_3 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = \mathsf{T}_{11} = \mathsf{T}_{12} = 1,$ $\mathsf{T}_{3+j} = j/6$ for $j = 1, \ldots, 5$
1.16	4	14	$\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}_2 = \mathsf{T}_3 = \mathsf{T}_4 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = \mathsf{T}_{11} = \mathsf{T}_{12} = \mathsf{T}_{13} = 1,$ $\mathsf{T}_{4+j} = j/5$ for $j = 1, \ldots, 4$

□

[Piegl and Tiller, 1997][5] recommend against use of equally spaced knots, which can lead to a singular $\mathbf{N}$ matrix (3.3) when used with either the chord length method (3.7) or the centripetal method (3.10). Instead, they recommend use of a so-called **averaging method**,

$$\left.\begin{aligned} &\mathsf{T}_0 = \cdots = \mathsf{T}_p = 0, \\ &\mathsf{T}_{p+j} = \frac{1}{p} \sum_{i=j}^{p-1+j} \tau_i, \ \text{for } j = 1, \ldots, (n-p), \\ &\mathsf{T}_{\mathtt{I}-p} = \cdots = \mathsf{T}_{\mathtt{I}} = 1. \end{aligned}\right\} \tag{3.13}$$

[5] At 365.

Example 28.

Averaging knots. Revisiting the examples shown in as shown in Figures. 1.11 through 1.16, we demonstrate evaluation of the knot vectors based on the averaging knots scheme in (3.13). Again, the number of sample points $(s+1)$ is set equal to the number of control points $(n+1)$, which is nine for all cases (thus $s = n = 8$). The sample time vector is $\boldsymbol{\tau} = \{\tau_k\}_{k=0}^{s} = \{\tau_k\}_{k=0}^{8}$.

Fig. #	degree p	# knots $(\mathtt{I}+1)$	knots $\mathsf{T}_0, \ldots, \mathsf{T}_\mathtt{I}$
1.11	1	11	$\mathsf{T}_0 = \mathsf{T}_1 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = 1,$ $\mathsf{T}_2 = \tau_1,\ \mathsf{T}_3 = \tau_2,\ \mathsf{T}_4 = \tau_3,\ \ldots,\ \mathsf{T}_8 = \tau_7$
1.12	2	12	$\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}_2 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = \mathsf{T}_{11} = 1,$ $\mathsf{T}_3 = \frac{1}{2}(\tau_1 + \tau_2),\ \mathsf{T}_4 = \frac{1}{2}(\tau_2 + \tau_3),\ \ldots,\ \mathsf{T}_8 = \frac{1}{2}(\tau_6 + \tau_7)$
1.15	3	13	$\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}_2 = \mathsf{T}_3 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = \mathsf{T}_{11} = \mathsf{T}_{12} = 1,$ $\mathsf{T}_4 = \frac{1}{3}(\tau_1 + \tau_2 + \tau_3),$ $\mathsf{T}_5 = \frac{1}{3}(\tau_2 + \tau_3 + \tau_4),\ \ldots,$ $\mathsf{T}_8 = \frac{1}{3}(\tau_5 + \tau_6 + \tau_7)$
1.16	4	14	$\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}_2 = \mathsf{T}_3 = \mathsf{T}_4 = 0,\quad \mathsf{T}_9 = \mathsf{T}_{10} = \mathsf{T}_{11} = \mathsf{T}_{12} = \mathsf{T}_{13} = 1,$ $\mathsf{T}_5 = \frac{1}{4}(\tau_1 + \tau_2 + \tau_3 + \tau_4),$ $\mathsf{T}_6 = \frac{1}{4}(\tau_2 + \tau_3 + \tau_4 + \tau_5),\ \ldots,$ $\mathsf{T}_8 = \frac{1}{4}(\tau_4 + \tau_5 + \tau_6 + \tau_7)$

□

Example 29.

Interpolation of five 2D points, $\{\tilde{\boldsymbol{P}}_i\}_{i=0}^{4} = \{\ (0,0),\ (3,4),\ (-1,4),\ (-4,0),\ (-4,-3)\ \}$ (orange '$+$' with diamond outline) using a cubic ($p = 3$) B-spline curve. $\square$

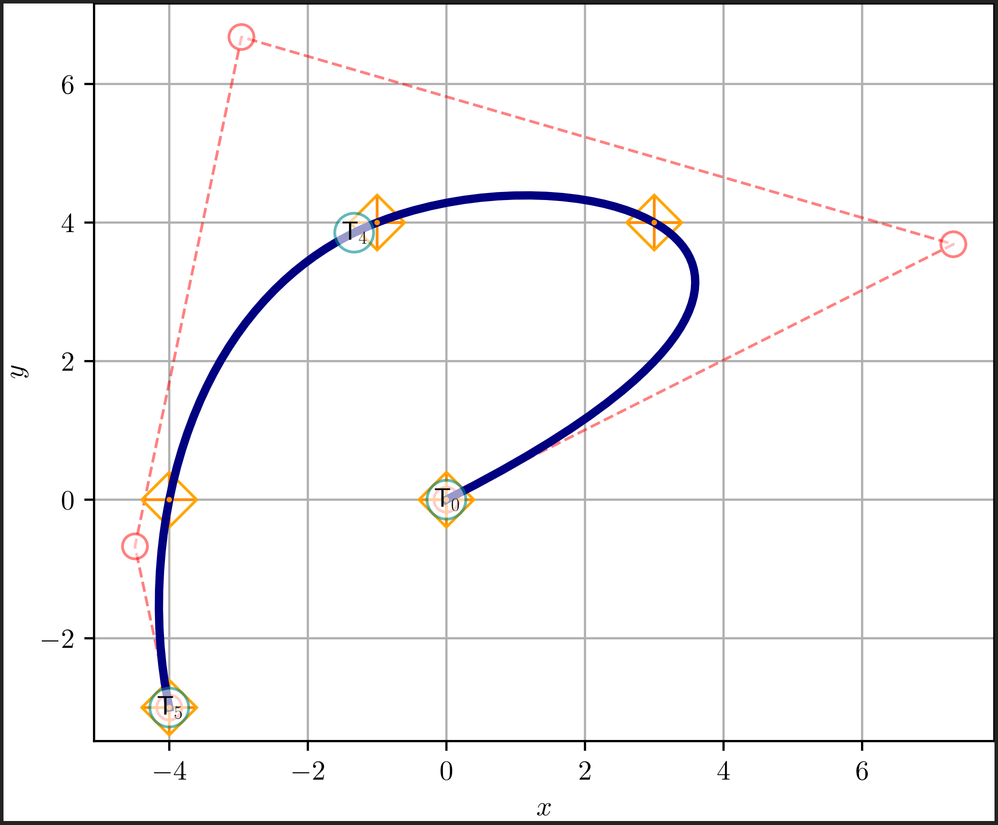

Figure 3.1: Reproduction of [Piegl and Tiller, 1997] Example 9.1. Source code on GitHub.

3.1.3 The Approximation General Case ($s > n$)

See [Piegl and Tiller, 1997], “Least Squares Curve Approximation” pages 410–412 for details.

Chapter 4

NURBS

Prior to defining the B-spline basis functions, we examined uniform knot vectors in Section 1.3 and then compared these to non-uniform knot vectors in Section 1.5. In this chapter, we start with the open knot vector (non-periodic, non-uniform), previously described in (1.18),

$$\mathbf{T} = \{\, \underbrace{\mathsf{T}_a, \ldots, \mathsf{T}_a}_{p+1},\ \mathsf{T}_{p+1},\ \ldots,\ \mathsf{T}_{m-p-1},\ \underbrace{\mathsf{T}_b, \ldots, \mathsf{T}_b}_{p+1}\,\}, \tag{4.1}$$

as our knot vector of choice. With this non-uniform knot vector, we create the original B-spline basis function $N_i^p(t)$ in combination with positive weighting constants to give a new *rational* basis function $R_i^p(t)$.

Conceptually, we will weight a single B-spline basis function by a positive number and then normalize this weighted B-spline quantity by the inner product of all original basis functions with all their respective weights. This creates a *rational* basis function. Mathematically, we define the i^{th} **rational basis function** as

$$R_i^p(t) \triangleq \frac{N_i^p(t)\ w_i}{\sum_{k=0}^{n} N_k^p(t)\ w_k}, \qquad \text{for } w_i \in \mathbb{R}^+, w_k \in \mathbb{R}^+ \tag{4.2}$$

where $N_i^p(t)$ and $N_k^p(t)$ denote a B-spline normalized basis function defined in (1.6) through (1.8), and all **weights** w_i and $\{w_k\}_{k=0}^n$ are positive, real numbers.

The combination of the non-uniform knot vector and the rational basis functions gives rise to the NURB acronym, <u>n</u>on-<u>u</u>niform, <u>r</u>ational <u>B</u>-spline. We define the NURB curve as

$$\mathbb{C}^p(t) \triangleq \sum_{i=0}^{n} R_i^p(t)\ \boldsymbol{P}_i, \qquad \text{for } t \in \mathbb{R} \subset [\ \mathsf{T}_0,\ \mathsf{T}_{\mathtt{I}}\ ], \tag{4.3}$$

where $\{\boldsymbol{P}_i\}_{i=0}^n$ are the $(n+1)$ control points. Compare the form of (4.3) to the form of (5.7).

Chapter 5

B-Spline Surfaces and Volumes

The B-Spline formulation for curves can be generalized to surfaces and volumes. Let the following knot vectors be defined as

$$\mathbf{T} = \{\underbrace{\mathsf{T}_a, \ldots, \mathsf{T}_a}_{p+1}, \ \mathsf{T}_{p+1}, \ \ldots, \ \mathsf{T}_{\mathtt{I}-p-1}, \ \underbrace{\mathsf{T}_b, \ldots, \mathsf{T}_b}_{p+1}\}, \tag{5.1}$$

$$\mathbf{U} = \{\underbrace{\mathsf{U}_a, \ldots, \mathsf{U}_a}_{q+1}, \ \mathsf{U}_{q+1}, \ \ldots, \ \mathsf{U}_{\mathtt{J}-q-1}, \ \underbrace{\mathsf{U}_b, \ldots, \mathsf{U}_b}_{q+1}\}, \tag{5.2}$$

$$\mathbf{V} = \{\underbrace{\mathsf{V}_a, \ldots, \mathsf{V}_a}_{r+1}, \ \mathsf{V}_{r+1}, \ \ldots, \ \mathsf{V}_{\mathtt{K}-r-1}, \ \underbrace{\mathsf{V}_b, \ldots, \mathsf{V}_b}_{r+1}\}. \tag{5.3}$$

5.1 Knot Dependence on Degree and Control Points

A B-spline basis function of degree p, q, and r with $(n+1)$, $(m+1)$, and $(l+1)$ control points will require $\mathtt{I}$, $\mathtt{J}$, and $\mathtt{K}$ knot spans and thus $(\mathtt{I}+1)$, $(\mathtt{J}+1)$, and $(\mathtt{K}+1)$ knots, where

$$\mathtt{I} = p + n + 1, \tag{5.4}$$

$$\mathtt{J} = q + m + 1, \tag{5.5}$$

$$\mathtt{K} = r + l + 1. \tag{5.6}$$

The number of knots has dependence on degree and control points as stated in Table 5.1.

Table 5.1: Knot number dependence on degree and control points.

(# knots)	=	(degree + 1)	+	(# control points)
(# knots)	=	(order)	+	(# control points)
$(\mathtt{I}+1)$	=	$(p+1)$	+	$(n+1)$
$(\mathtt{J}+1)$	=	$(q+1)$	+	$(m+1)$
$(\mathtt{K}+1)$	=	$(r+1)$	+	$(l+1)$

5.2 Generalized B-Splines Geometries

Then with the control points (generally in 3D) $\boldsymbol{P}_i(x,y,z)$, $\boldsymbol{P}_{i,j}(x,y,z)$, and $\boldsymbol{P}_{i,j,k}(x,y,z)$, arranged into a collection (array in 1D, net/grid in 2D, lattice in 3D) to describe a B-Spline object in 1D, 2D, and 3D, respectively, the B-Spline curve, surface, and volume are defined as

$$\mathbb{C}^p(t) \triangleq \sum_{i=0}^{n} N_i^p(t)\ \boldsymbol{P}_i, \tag{5.7}$$

$$\mathbb{S}^{p,q}(t,u) \triangleq \sum_{i=0}^{n}\sum_{j=0}^{m} N_i^p(t)\ N_j^q(u)\ \boldsymbol{P}_{i,j}, \tag{5.8}$$

$$\mathbb{V}^{p,q,r}(t,u,v) \triangleq \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l} N_i^p(t)\ N_j^q(u)\ N_k^r(v)\ \boldsymbol{P}_{i,j,k}, \tag{5.9}$$

for

$$t \in \mathbb{R} \subset [\ \mathsf{T}_a,\ \mathsf{T}_b\ ], \tag{5.10}$$

$$u \in \mathbb{R} \subset [\ \mathsf{U}_a,\ \mathsf{U}_b\ ], \tag{5.11}$$

$$v \in \mathbb{R} \subset [\ \mathsf{V}_a,\ \mathsf{V}_b\ ]. \tag{5.12}$$

Example 30.

B-Spline surface construction. Twelve 3D control points, $[\{\ \boldsymbol{P}_{i,j}\ \}_{j=0}^{2}\ ]_{i=0}^{3}$, organized into a control net shown in Table 5.2, with $(n+1) = 4$ control points for the t parameter and $(m+1) = 3$ control points for the u parameter. A cubic ($p = 3$) B-spline curve is used for the t parameter space. A quadratic ($q = 2$) B-spline curve is used for the u parameter space. Thus, the number of knots is

$$(\mathtt{I}+1) = (p+1)+(n+1) = (3+1)+(4) = 8, \tag{5.13}$$

$$(\mathtt{J}+1) = (q+1)+(m+1) = (2+1)+(3) = 6, \tag{5.14}$$

and the knot vectors are

$$\mathbf{T} = \{\ \underbrace{0,\ 0,\ 0,\ 0}_{p+1=4},\ \underbrace{1,\ 1,\ 1,\ 1}_{p+1=4}\ \}. \tag{5.15}$$

$$\mathbf{U} = \{\ \underbrace{0,\ 0,\ 0}_{q+1=3},\ \underbrace{1,\ 1,\ 1}_{q+1=3}\ \}. \tag{5.16}$$

Note that the structure of these B-spline knot vectors recovers a Bézier surface patch (see Eq. 1.28). □

Table 5.2: Control points $\boldsymbol{P}_{i,j}(x, y, z)$, arranged into a control net.

	$j = 0$	$j = 1$	$j = 2$
$i = 0$	(0, 0, 0)	(0, 4, 0)	(0, 8, -3)
$i = 1$	(2, 0, 6)	(2, 4, 0)	(2, 8, 0)
$i = 2$	(4, 0, 0)	(4, 4, 0)	(4, 8, 3)
$i = 3$	(6, 0, 0)	(6, 4, -3)	(6, 8, 0)

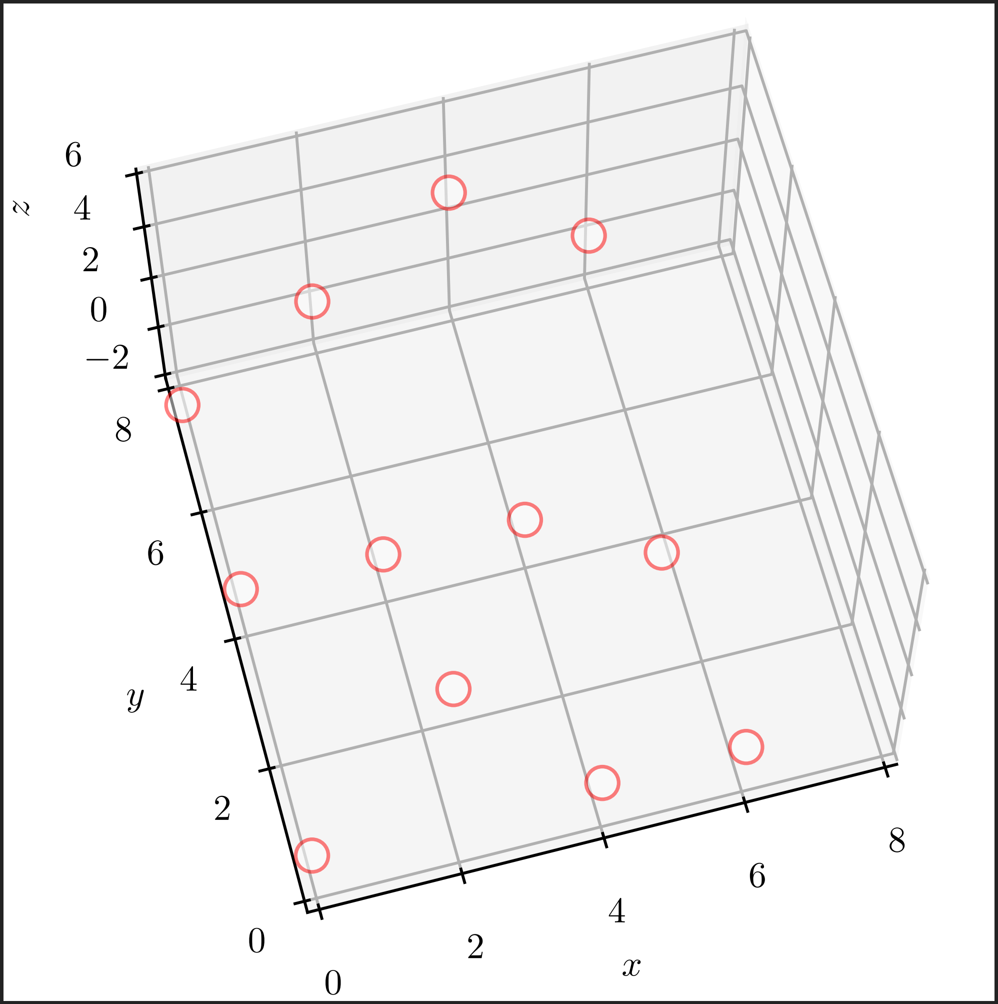

Figure 5.1: Twelve control points, located at their (x, y, z) coordinates listed in Table 5.2. Reproduction of [Bingol and Krishnamurthy, 2019] 3D surface. Source code on GitHub.

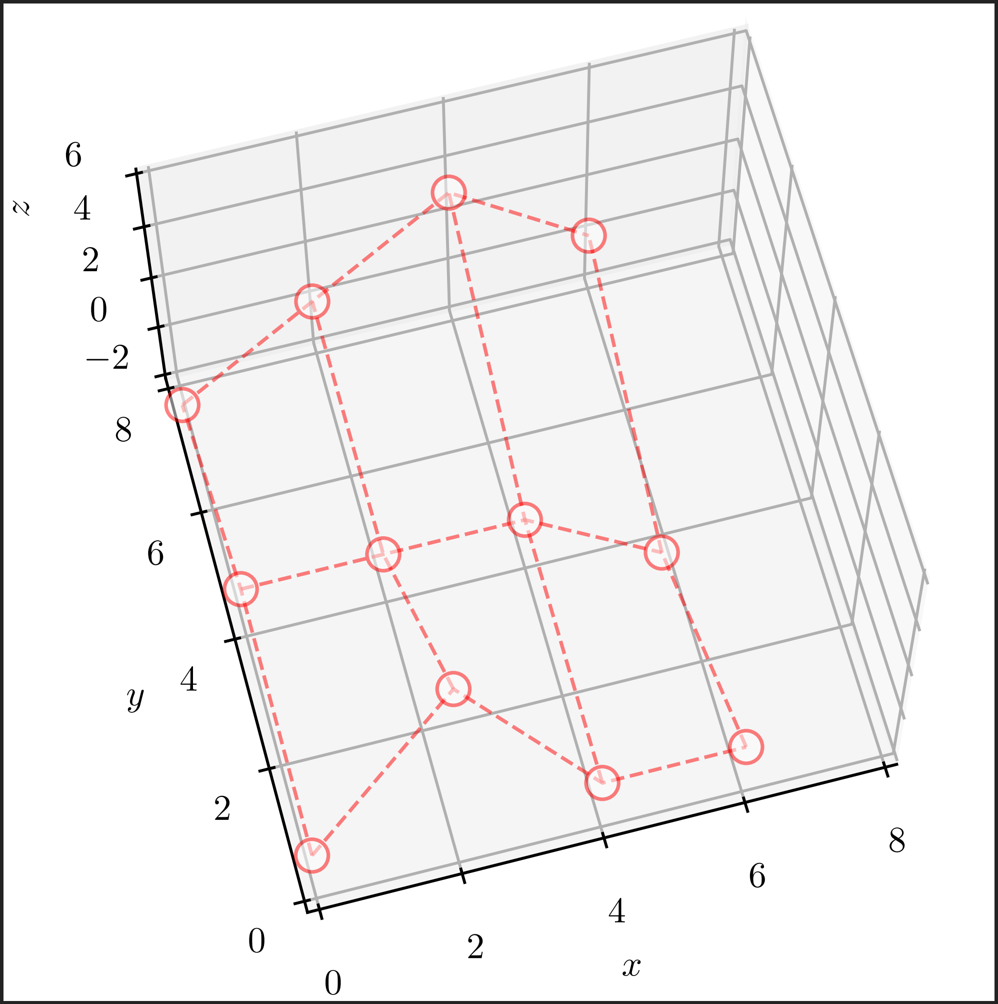

Figure 5.2: *Continued from previous figure.* Single B-spline surface control net, connecting the twelve control points.

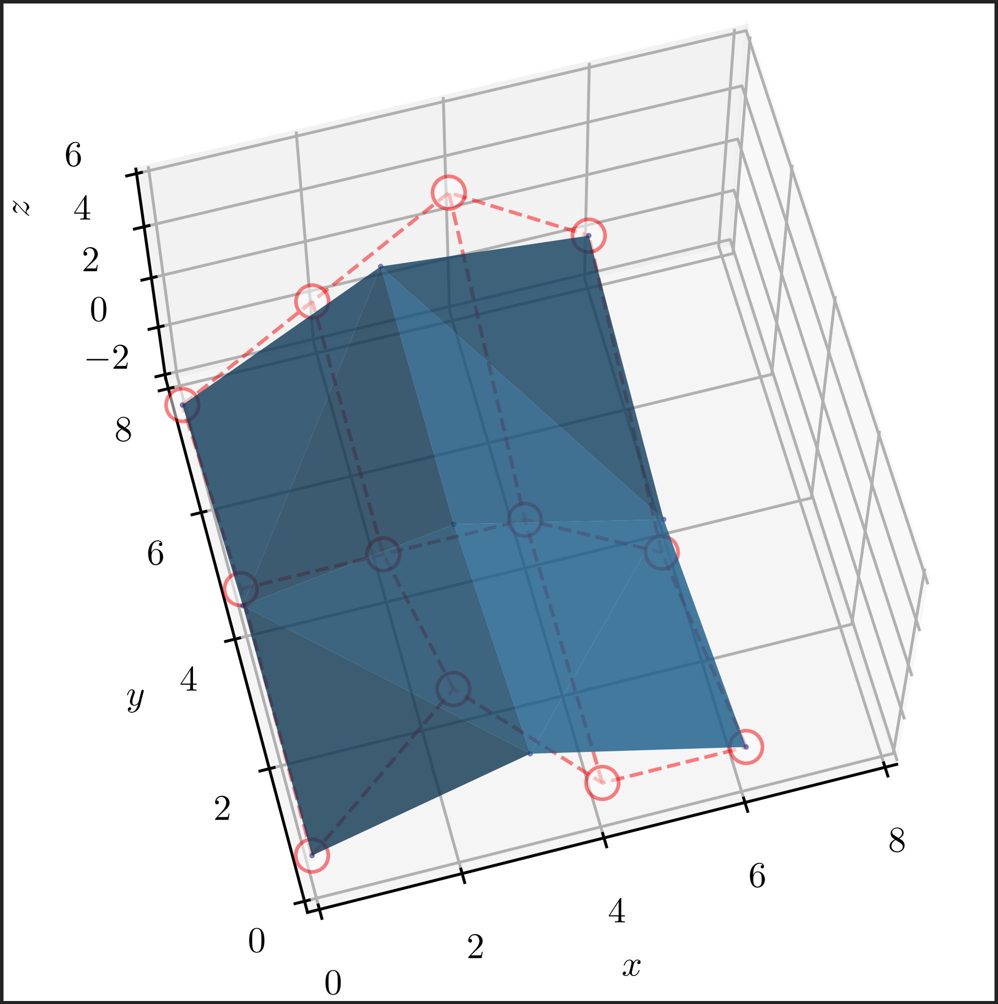

Figure 5.3: *Continued from previous figure.* Single B-spline surface patch, with a single bisection evaluation ($2^1 = 2$ evaluation intervals) of the knot vectors $\mathbf{T}$ and $\mathbf{U}$.

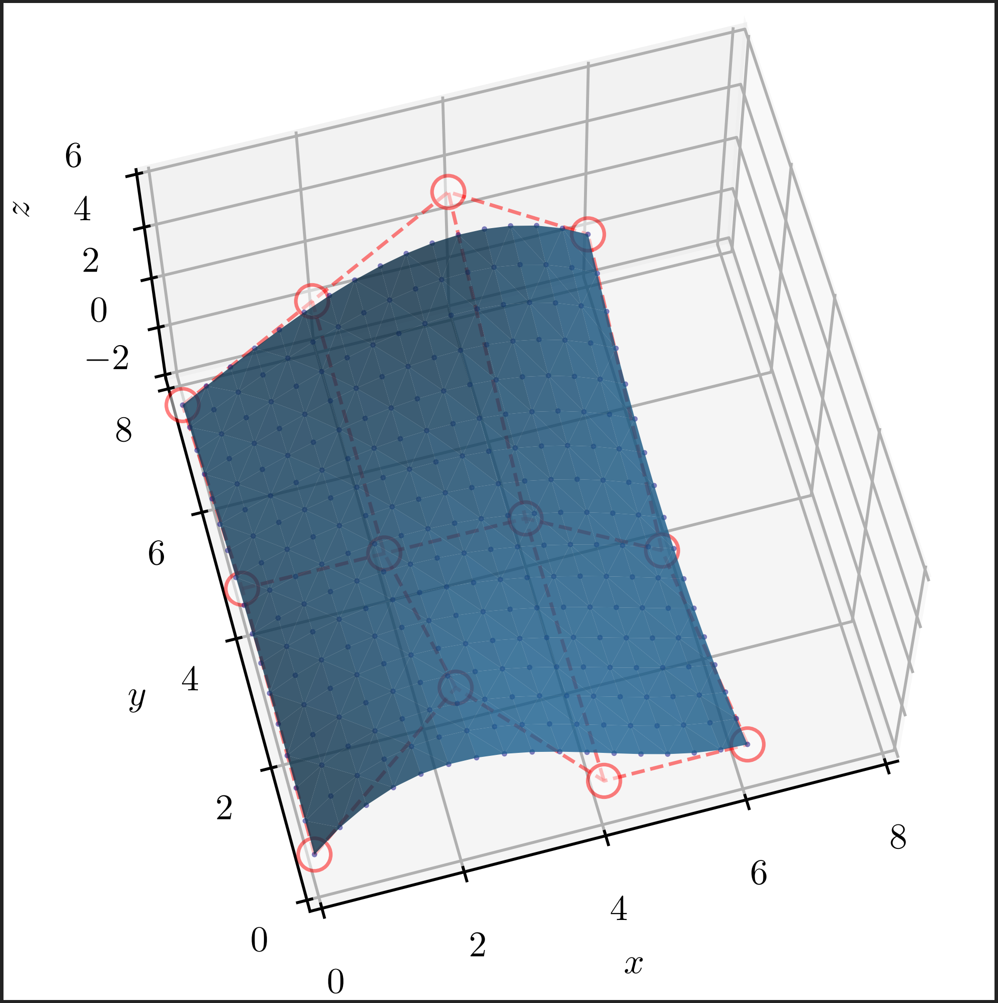

Figure 5.4: *Continued from previous figure.* Single B-spline surface patch, with four bisection evaluations ($2^4 = 16$ evaluation intervals) of the knot vectors $\mathbf{T}$ and $\mathbf{U}$.

5.3 Shape Primitives

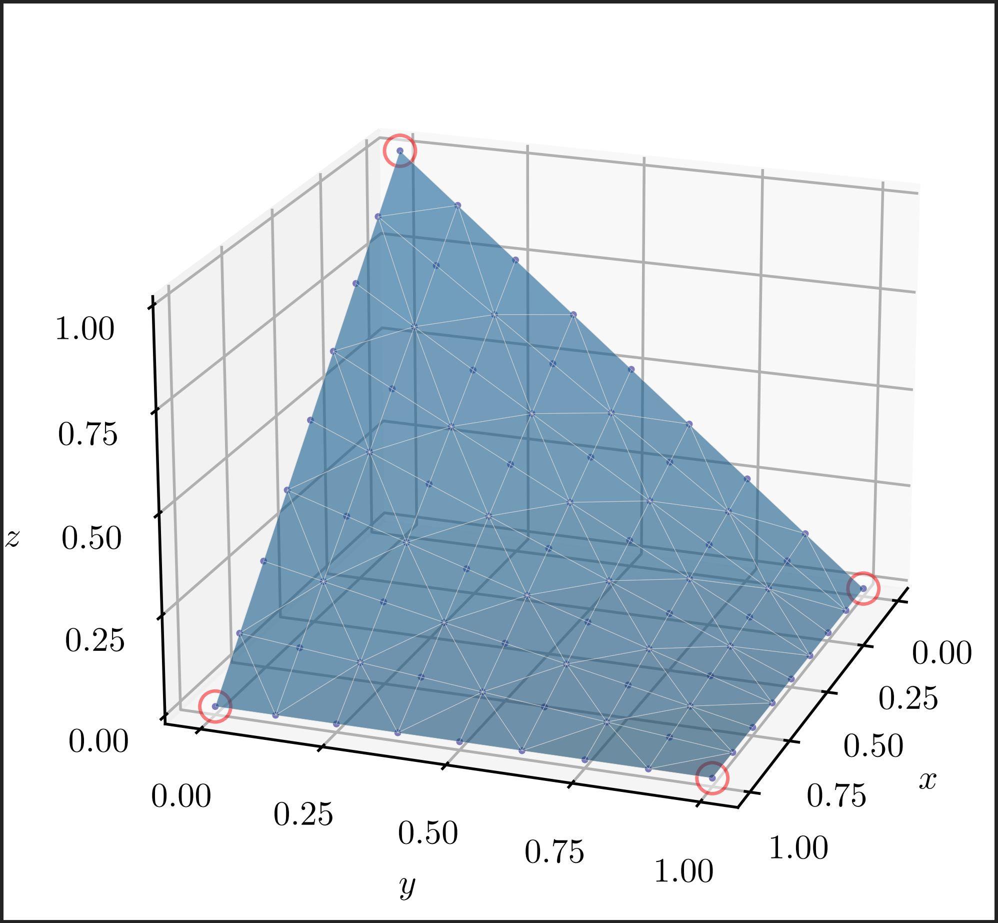

Figure 5.5: Recovery of the first Bézier bi-linear shape function. Red circles are control points. Blue dots are surface evaluation points. Source code `bspline_surface_Bezier_recovery.py` on GitHub.

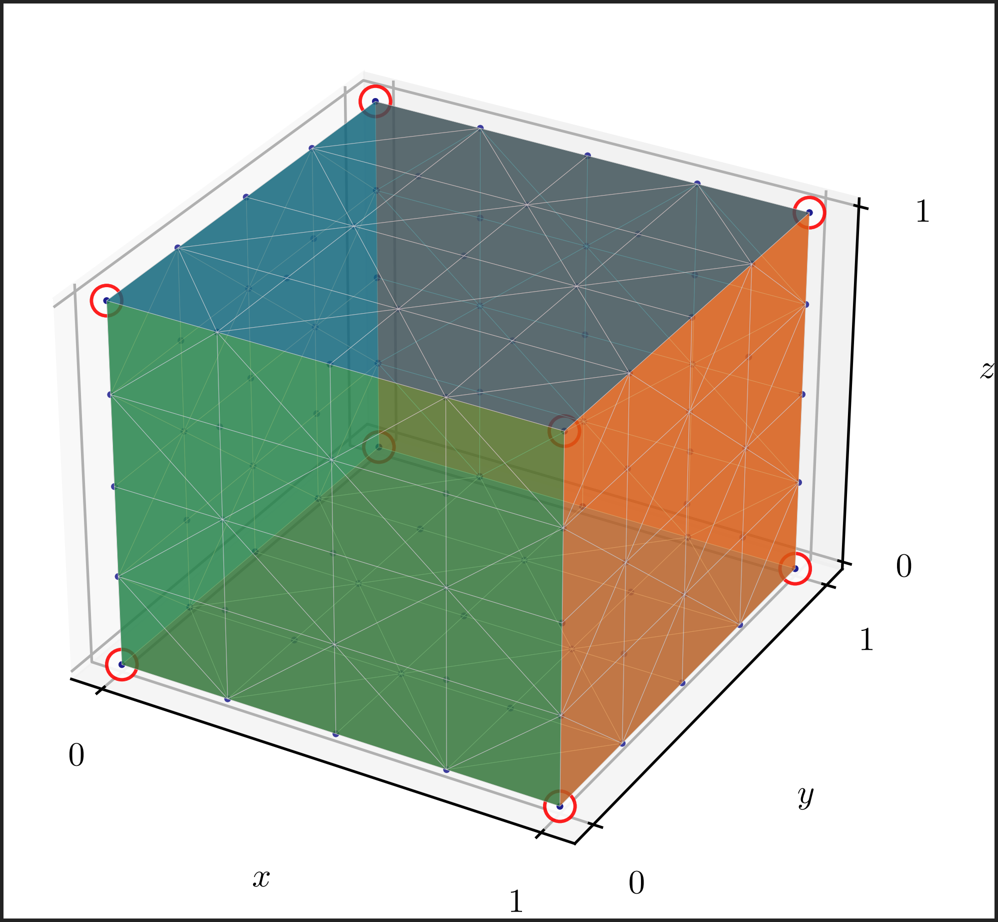

Figure 5.6: A tri-linear cube composed of six bi-linear surfaces. Red circles are control points. Blue dots are surface evaluation points. Source code `bspline_surface_cube_linear.py` on GitHub.

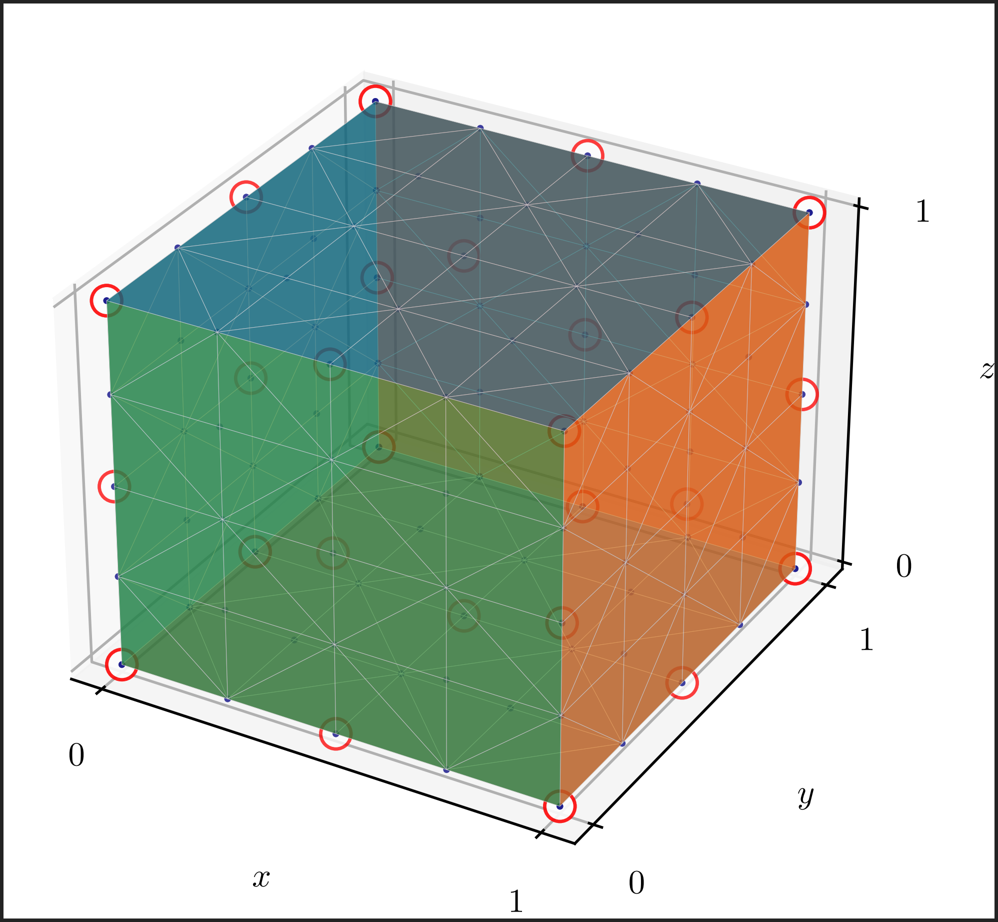

Figure 5.7: A tri-quadratic cube composed of six bi-quadratic surfaces. Red circles are control points. Blue dots are surface evaluation points. Source code `bspline_surface_cube_quadratic.py` on GitHub.

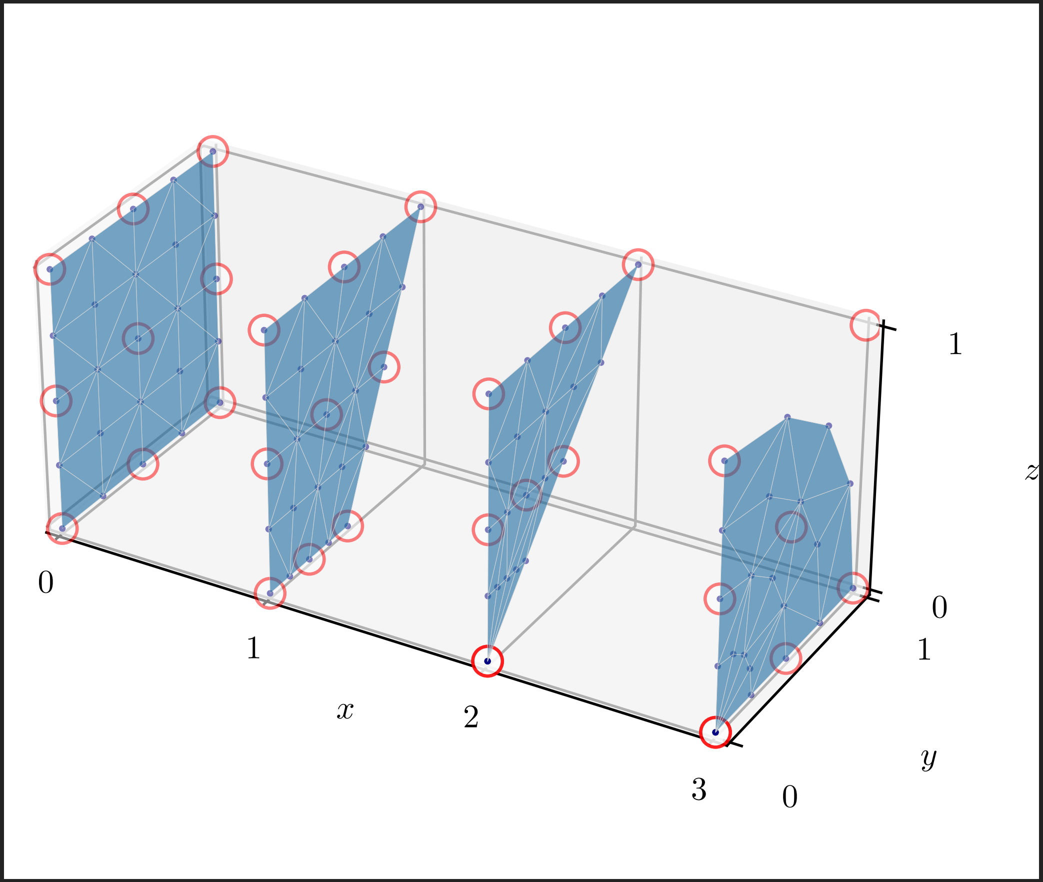

Figure 5.8: Transformation of a bi-quadratic surface ($x = 0$) into a bi-quadratic quarter-cylinder end cap ($x = 3$) using $(0, y, 0)$ control point coalescence to $(0, 0, 0)$ and perimeter control point rebalancing. Source code `bspline_surface_quad2tri_quadratic.py` on GitHub.

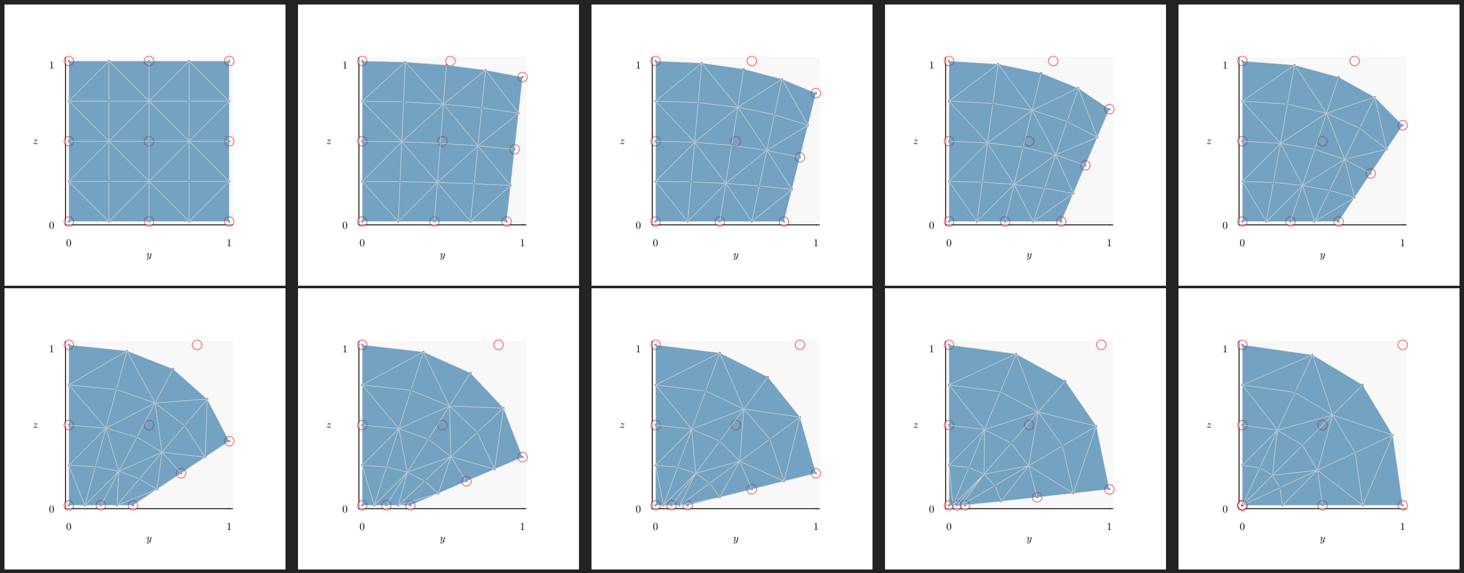

Figure 5.9: *Continued from the previous figure.* Planar view sequence of transformation from quadrilateral perimeter to triangular perimeter. Source code `bspline_surface_biquad2tri_animation.py` on GitHub.

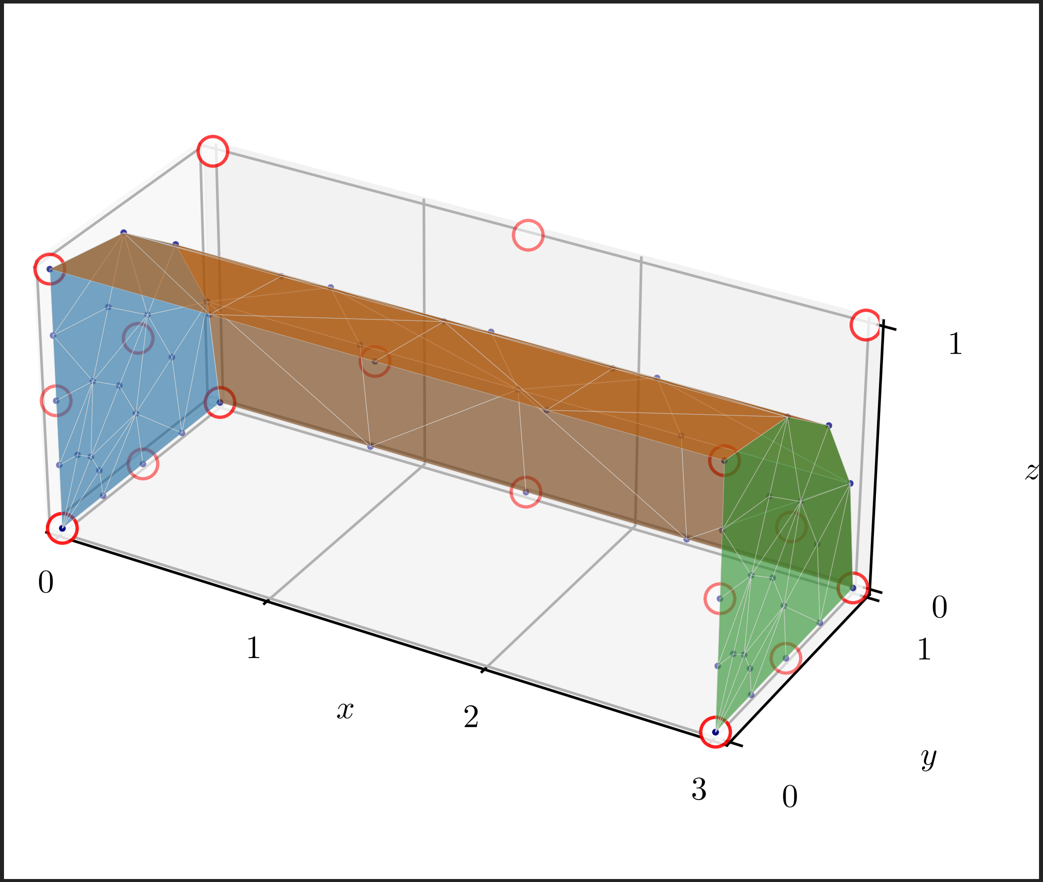

Figure 5.10: A tri-quadratic cylinder composed of three bi-quadratic surfaces. Red circles are control points. Blue dots are surface evaluation points. Source code `bspline_surface_qtrcyl_quadratic.py` on GitHub.

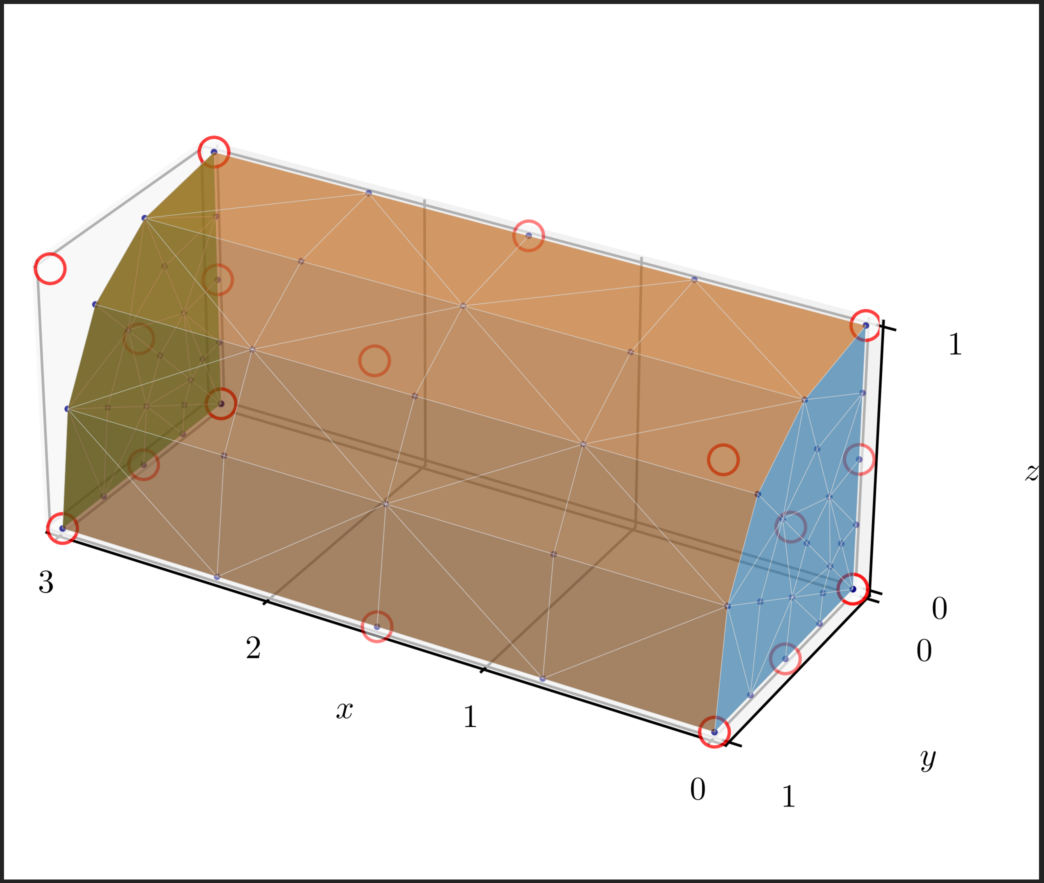

Figure 5.11: Alternative view of previous figure, a tri-quadratic cylinder.

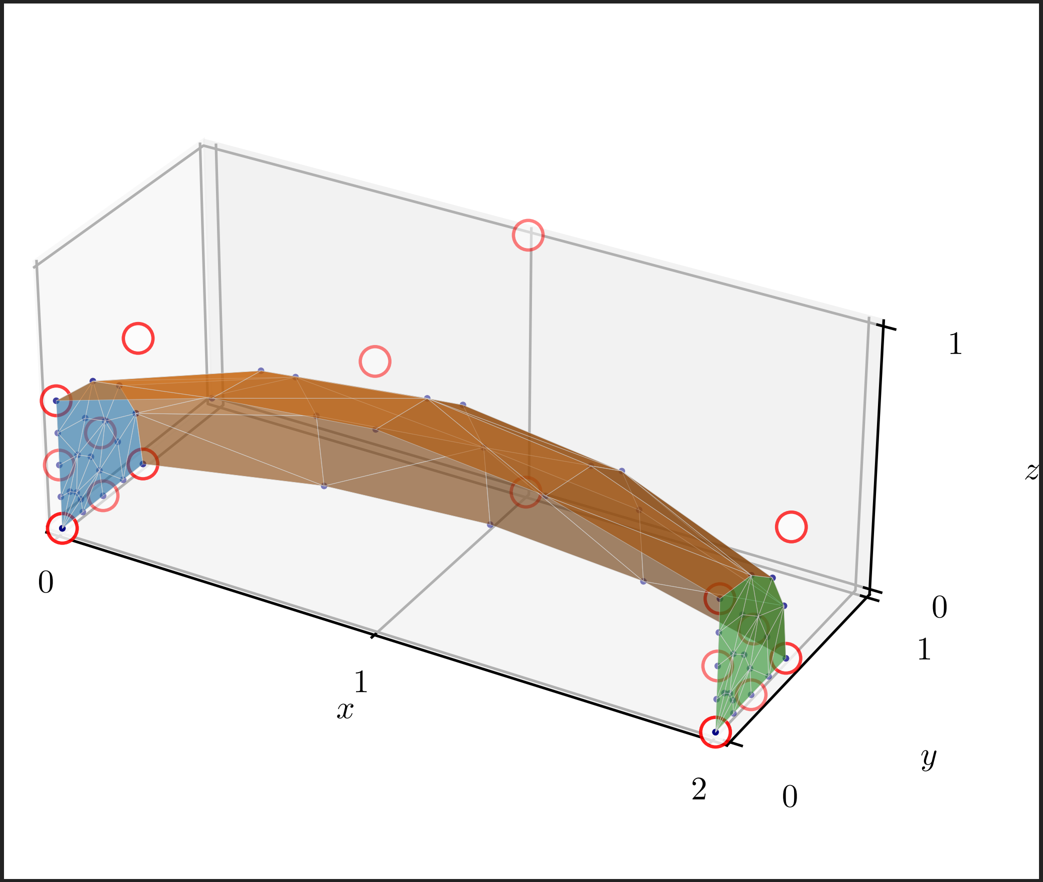

Figure 5.12: A tri-quadratic cylinder morphed toward a sphere through pole coalescence. Source code `bspline_surface_qtrcyl2sphere_quadratic.py` on GitHub.

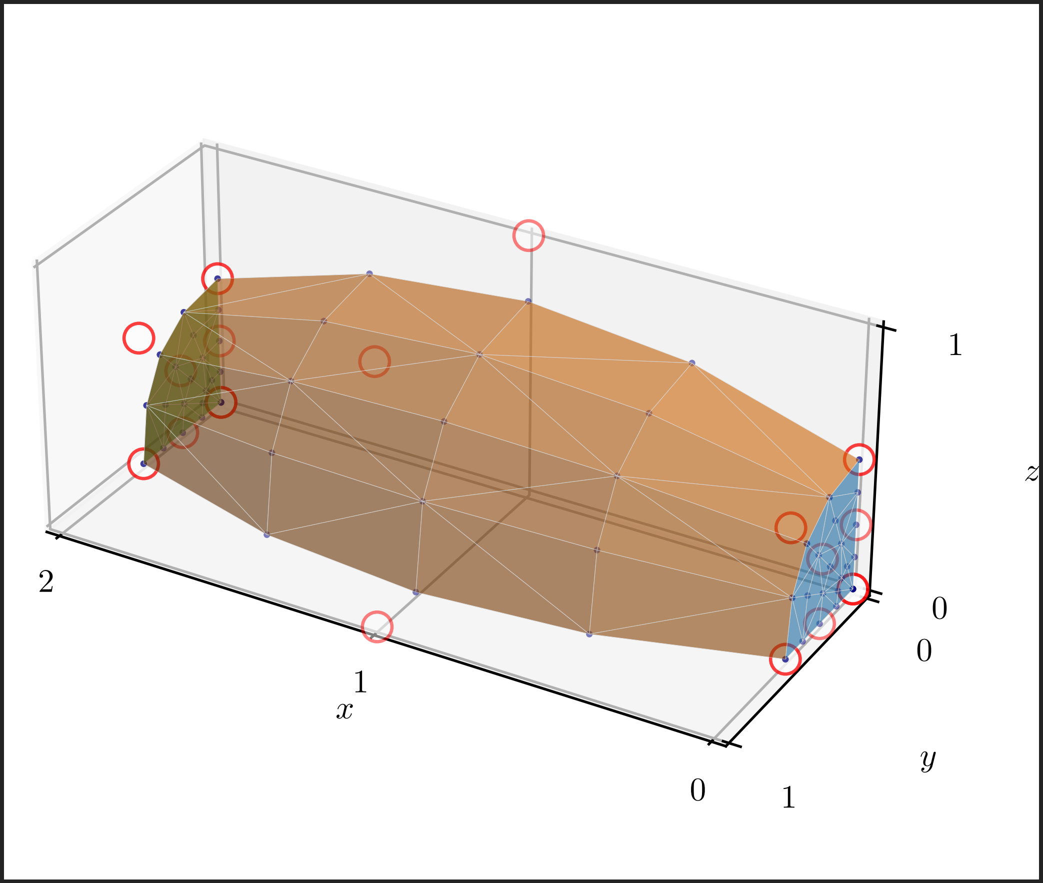

Figure 5.13: Alternative view of previous figure, a tri-quadratic cylinder morphing toward a sphere.

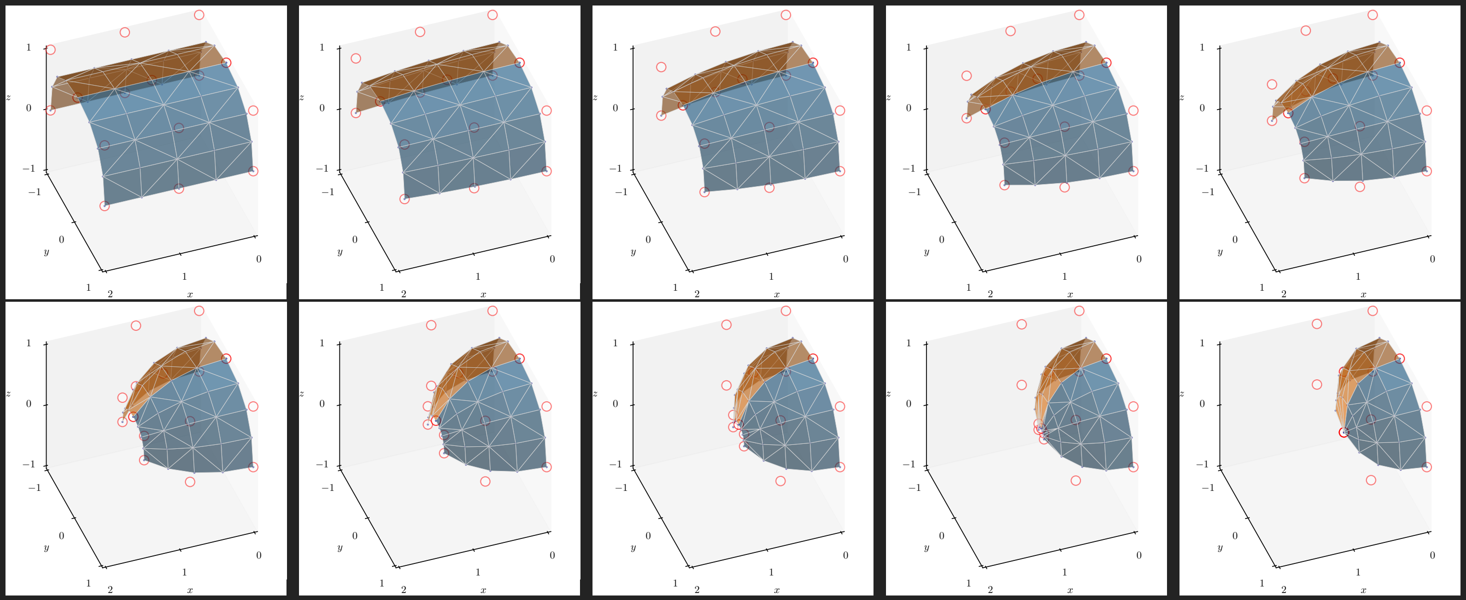

Figure 5.14: Isometric view sequence of transformation from half-cylinder to half-sphere. Source code `bspline_surface_cyl2sphere_animation.py` on GitHub.

Chapter 6

Acknowledgements

We gratefully acknowledge support from the Office of Naval Research (Dr. Timothy Bentley) under Special Studies grant N0001418IP00054.

Bibliography

[Bingol and Krishnamurthy, 2019] Bingol, O. R. and Krishnamurthy, A. (2019). NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX*, 9:85–94.

[Cottrell et al., 2009] Cottrell, J. A., Hughes, T. J., and Bazilevs, Y. (2009). *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons.

[Eberly, 2020] Eberly, D. (2020). Least-squares fitting of data with b-spline curves. *Geometric Tools*, pages 1–5. https://www.geometrictools.com/Documentation/BSplineCurveLeastSquaresFit.pdf.

[Piegl and Tiller, 1997] Piegl, L. and Tiller, W. (1997). *The NURBS book*. Springer Science & Business Media.