ISSN 2464-4617 (print)
ISSN 2464-4625 (online) Computer Science Research Notes - CSRN http://www.wscg.eu

Projective Geometric Algebra powers up Bézier Curves

Enzo Harquin[1] Stéphane Breuils[2] Venceslas Biri[1] Vincent Nozick[1]

[1] Université Gustave Eiffel, LIGM, Champs-sur-Marne, F-77420, France
{enzo.harquin2,venceslas.biri,vincent.nozick}@univ-eiffel.fr
[2] Université Savoie Mont-Blanc, CNRS, LAMA, Chambéry, F-73000, France
stephane.breuils@univ-smb.fr

ABSTRACT

Bézier curves are a cornerstone of computer-aided geometric design, yet their classical formulations operate within Euclidean or projective function spaces. We show that rational Bézier curves arise intrinsically from the algebraic structure of PGA (Projective Geometric Algebra), providing a unified, coordinate-free treatment that naturally extends classical constructions by allowing negative weights and points at infinity. Negative weights may create singularities that correspond to points at infinity, admitting a consistent geometric interpretation. This enables a natural representation of complementary curve sections and conic types. We then generalize the Bézier construction beyond points to higher geometric objects of PGA, including lines, planes, and higher-dimensional primitives. Finally, we extend the framework to versors, the algebraic encodings of rigid body transformations in PGA, yielding smooth Bézier curves of such transformations, suitable for geometric animation. All constructions share the same blending equations and normalization procedure, demonstrating that PGA provides a genuinely unified foundation for curve modeling across geometric objects and transformations.

Keywords

Bézier Curves, Projective Geometric Algebra, Geometric Algebra, Parametric Curves

1 INTRODUCTION

Bézier curves were introduced in the 1960s by Pierre Bézier while working at Renault [BÉZ92], and independently by Paul de Faget de Casteljau at Citroën [DC85]. These curves quickly became fundamental in computer-aided design (CAD) and computer graphics due to their intuitive control point representation and smooth interpolation properties. Subsequently, cubic splines, notably formalized by de Boor [dB78], emerged as a powerful tool for smooth curve interpolation, particularly suited for ensuring C^2-continuity across piecewise polynomial segments, making them widely used in data fitting, animation, and trajectory planning. Over time, these concepts were extended to Rational Bézier curves and to B-splines [Far02], introducing rational formulations for the exact representation of conic sections and spline-based constructions providing local control and adjustable smoothness over arbitrary-degree curves. These developments culminated in Non-Uniform Rational B-Splines (NURBS), now the industry standard in CAD and computer graphics, offering a flexible and powerful framework for modeling curves and surfaces. For more details, the reader can refer to [PT12, FH00].

Despite their success, spline-based representations are formulated primarily in polynomial or rational function spaces and are mainly designed to interpolate points. Representing and manipulating higher level geometric entities, such as lines, planes, or conic sections, typically requires ad hoc constructions. Moreover, while rational curves exploit homogeneous coordinates, their interaction with projective geometry is handled externally rather than intrinsically, so operations like transformations, intersections, and dual entities are not native to the representation.

In contrast, geometric algebra (GA), first formalized by William Kingdon Clifford in the late 19th century based on earlier work by Grassmann and Hamilton, provides a unified, coordinate-free algebraic framework for representing geometric objects and their transformations. Within a single algebraic structure, GA naturally encodes points, lines, planes, and higher-dimensional entities, while offering compact and expressive formulations of operations such as rotations, reflections, and rigid motions. For more details, the reader can refer to [DFM07]. Over the last decades, GA has gained increasing attention in physics, robotics, computer graphics, and computer-aided geometric design due to its conceptual clarity and computational efficiency [Per09, DL03, Som13]. In particular, Projective Geometric Algebra (PGA) [Gun17, DK20], has emerged as a powerful modern framework unifying Euclidean and projective geometry, enabling elegant representations of rigid body motions, intersections, and geometric transformations within a single algebra. More recently, GA has also found applications in deep

learning, where GA-based architectures naturally preserve geometric equivariances under rotations and rigid transformations [RGDK+23, BDHBC23].

Recently, Hugo Hadfield [Had23] addressed related questions in his doctoral thesis by investigating direct linear interpolation of geometric objects in conformal geometric algebra, while briefly discussing Bézier curves and Hermite splines construction from geometric primitives. While these works demonstrate the expressive power of geometric algebra for curve modeling, a unified and systematic treatment of Bézier curves within a projective geometric algebra framework, particularly one that incorporates weighted control elements, ideal-points, and mixed geometric entities, remains an open research direction, which this paper aims to address.

The contributions of this paper are the following. At first, we extend the classical Bézier framework to PGA, including negative weights, giving a consistent geometric interpretation to singularities as points at infinity. We show that finite points, ideal points, and mixed configurations are handled uniformly by the same blending equations (Sec. 4). Secondly, we generalize Bézier interpolation to arbitrary geometric primitives, including lines and planes, and more generally hyperplanes of any dimension. We extend the construction to yield smooth Bézier curves of rigid transformations (Sec. 5).

2 PGA AND NOTATION

Projective Geometric Algebra (PGA), introduced by Charles Gunn [Gun17], provides a unified framework to represent points, lines, planes, hyperplanes and transformations in projective space. In the *plane-based formulation* of PGA, hyperplanes are treated as the fundamental elements, and other geometric entities (such as lines, points, etc.) are obtained from the outer product (intersection) of hyperplanes [DFM07, Per09]. This approach naturally handles homogeneous coordinates and rigid transformations in a coordinate-free manner.

PGA can be used in any n-dimensional space, where grade-1 elements are nd hyperplanes. It is commonly denoted $\mathbb{R}_{n,0,1}$ in the literature, meaning that n basis vectors square to $+1$ and one basis vector squares to 0, namely $\mathbf{e}_0^2 = 0$, $\mathbf{e}_1^2 = \cdots = \mathbf{e}_n^2 = 1$, and $\mathbf{e}_i \cdot \mathbf{e}_j = 0$ for $i \neq j$. Illustrations in this paper are mostly given in dimension $n = 2$, where hyperplanes reduce to lines, though all presented constructions extend to any dimension. We adopt the following notation conventions:

- **1-blades** (homogeneous multivectors of grade-1) are denoted by *lower-case bold letters*, e.g., $\mathbf{a}$, $\mathbf{b}$. In plane-based PGA, these represent nd hyperplanes (including the ideal hyperplane at infinity).
- k**-blades** of grade $k > 1$ are denoted by *upper-case bold letters*, e.g., $\mathbf{A}$, $\mathbf{B}$. Blades correspond to subspaces such as lines (intersection of two planes) or points (intersection of three planes) in 3d PGA.
- **Multivectors** (general linear combinations of blades of different grades) are denoted by *upper-case non-bold letters*, e.g., A, B.

Formally, an nd hyperplane element (1-blade) can be expressed as a linear combination of canonical basis hyperplanes:

$$\mathbf{a} = a_0\mathbf{e}_0 + a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + \cdots + a_n\mathbf{e}_n, \tag{1}$$

where $\mathbf{e}_0$ represents the ideal hyperplane (hyperplane at infinity), and $\mathbf{e}_1, \ldots, \mathbf{e}_n$ correspond to Euclidean coordinate hyperplanes. With $a_0, \cdots a_n$ corresponding exactly to the Hessian form of an nd hyperplane $a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$.

PGA objects are either finite, with a well-defined Euclidean position, or ideal, lying at infinity and encoding a direction. This distinction is central to the norm definitions that follow.

Multivector Norms

The norm $\|A\|_f$ of a finite object A is defined uniformly by using the reverse

$$\|A\|_f = \sqrt{A\widetilde{A}}. \tag{2}$$

The reverse of A, denoted by $\widetilde{A}$, consists of an extension of the complex conjugate for multivectors. The reader can refer to [DDK22] for a complete explanation and the library `kingdon` [Roe25] for implementation purpose.

For ideal objects (e.g. point or line at infinity, etc.), the previous norm cancels because of the degenerate metric. For such a case we use a specific norm, denoted as $\|A\|_\infty$, built with the dual object finite norm

$$\|A\|_\infty = \|A^*\|_f. \tag{3}$$

See [Gun17] for further details.

The general norm, applicable to any PGA object, is then

$$\|A\| = \begin{cases} \|A\|_f & \text{if } A \text{ is finite,} \\ \|A\|_\infty & \text{if } A \text{ is ideal} \end{cases} \tag{4}$$

Let $\hat{A}$ be the normalized form of A, defined as

$$\hat{A} = \frac{A}{\|A\|} \tag{5}$$

so that $\|\hat{A}\| = 1$ by construction.

Duality and Points

In nd plane-based PGA, the dual of A, denoted as $A^\star$, provides a natural correspondence between hyperplanes of grade 1 and points of grade $n-1$, or more generally, between k-blades and $(n-k)$-blades. Specifically, a point can be defined as the dual of a hyperplane

$$\mathbf{P} = (w\mathbf{e}_0 + x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \cdots + x_n\mathbf{e}_n)^\star \tag{6}$$

where the x_i correspond to the Euclidean coordinates and w can be considered as an equivalent of the homogeneous coordinate of $\mathbb{P}^n$.

Points in PGA are classified as either finite points or ideal-points:

- A **finite point** has a nonzero homogeneous coordinate $w \neq 0$. Its norm, derived from Eq. (2), equals the absolute value of its homogeneous coordinate

$$\|\mathbf{P}\| = \|\mathbf{P}\|_f = \sqrt{\mathbf{P}\widetilde{\mathbf{P}}} = \sqrt{w^2} = |w| \tag{7}$$

  Normalizing a finite point from Eq. (5) satisfies $\|\hat{\mathbf{P}}\| = 1$, independently of $w = 1$ or $w = -1$.

- An **ideal-point** lies at infinity and has $w = 0$. Its norm, derived from Eq. (3), coincides with the standard L_2 vector norm, also called infinity norm in Sec.4.3 of [DDK22].

$$\|\mathbf{P}\| = \|\mathbf{P}\|_\infty = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{8}$$

Transformation Versor

In PGA, geometric transformations, like rotation, translation and reflection, are represented by versors, the reader can refer to [DFM07] for further details. A unit versor V acts on any object of the algebra $\mathbf{X}$ via the sandwich product

$$\mathbf{X} \mapsto V\,\mathbf{X}\,\widetilde{V}. \tag{9}$$

A unit versor satisfies $\|V\| = 1$, and is obtained by normalization via Eq. (5). It is worth noting that PGA versors provide a generalization of quaternions and dual quaternions to arbitrary dimensions.

3 BEZIER CURVES AND PROJECTIVE GEOMETRY

Bézier curves are traditionally formulated in Euclidean or projective space using control points associated with weights. In this section, we first recall the classical definitions of polynomial and rational Bézier curves and discuss their projective geometric interpretation, as a foundation for the subsequent reformulation within the PGA framework.

3.1 Polynomial Bézier Curves

In an n-dimensional (nd) space, Bézier curves, of degree d, are defined as weighted interpolations of a finite set of control points $P_i \in \mathbb{R}^n$ using the classical Bernstein basis polynomials $B_i^d(t)$

$$P(t) = \sum_{i=0}^{d} B_i^d(t)\,P_i, \qquad t \in [0,1] \tag{10}$$

where Bernstein basis polynomials are defined as

$$B_i^d(t) = \frac{d!}{i!(d-i)!} t^i (1-t)^{d-i}. \tag{11}$$

Since the Bernstein basis forms a partition of unity, $\sum_{i=0}^{d} B_i^d(t) = 1 \quad \forall t \in [0,1]$ and each basis polynomial satisfies $B_i^d(t) \geq 0$, each point $P(t)$ is a barycenter of the control points P_i with barycentric coordinates $B_i^d(t)$. This guarantees affine invariance, numerical stability, and the convex hull property: the curve remains inside the convex hull of its control points. An example is depicted in Fig. 1.

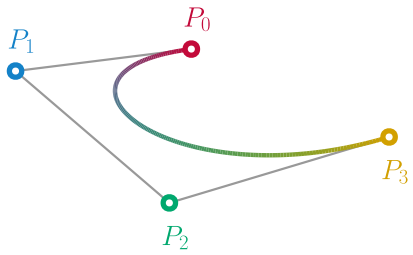

Figure 1: Cubic polynomial Bézier curve. The color gradient along the curve reflects the influence of each control point. The control polygon is shown in gray.

Moreover, the curve interpolates its endpoints, $P(0) = P_0$ and $P(1) = P_d$, since $B_0^d(0) = 1$ and $B_d^d(1) = 1$. In contrast, the intermediate control points act as transition points that govern the tangent directions and the global shape of the curve. In particular, the first segment P_0P_1 determines the initial tangent direction, while the last segment $P_{d-1}P_d$ determines the final tangent direction.

3.2 Rational Bézier Curves

Polynomial Bézier curves can be extended by associating a positive scalar weight $w_i > 0$ with each control point. This leads to rational Bézier curves, which allow the exact representation of a broader class of shapes, including conic sections (quadratic case) and other projective curves [Far02]. A rational Bézier curve is classically defined as

$$P(t) = \frac{\sum_{i=0}^{d} B_i^d(t)\,w_i P_i}{\sum_{i=0}^{d} B_i^d(t)\,w_i} \qquad t \in [0,1]. \tag{12}$$

An example of a rational Bézier curve and the influence of the weights is illustrated in Fig. 2.

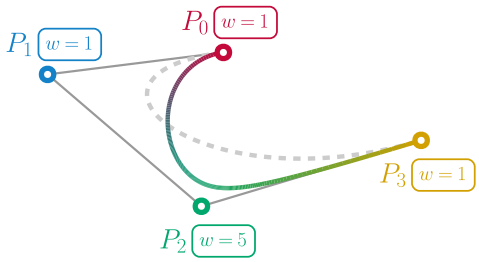

Figure 2: Cubic rational Bézier curve illustrating the influence of weights on the curve shape and the color gradient. The polynomial Bézier curve is shown with the dotted gray curve.

3.3 Projective Interpretation

From the viewpoint of projective geometry, rational Bézier curves are obtained by lifting control points into homogeneous coordinates. In this representation, each control point $P_i \in \mathbb{R}^n$ is expressed as $\mathrm{P}_i \in \mathbb{P}^n$

$$\mathrm{P}_i = (P_i, 1)^\top \doteq (kP_i, k)^\top \qquad k \neq 0, \tag{13}$$

where $\doteq$ refers to an equivalence up to non zero scale. In projective space, the curve is first expressed as

$$\mathrm{P}(t) = \sum_{i=0}^{d} B_i^d(t)\, w_i \mathrm{P}_i = \left(\sum_{i=0}^{d} B_i^d(t)\, w_i P_i,\ \sum_{i=0}^{d} B_i^d(t)\, w_i \right)^\top. \tag{14}$$

Despite common practice advising against adding points in $\mathbb{P}^n$, the sum of two appropriately weighted normalized finite points behaves like a barycentre

$$\begin{aligned}(k_1P_1, k_1)^\top + (k_2P_2, k_2)^\top &= (k_1P_1 + k_2P_2, k_1 + k_2)^\top \\ &\doteq \left(\frac{k_1P_1 + k_2P_2}{k_1 + k_2}\right)^\top.\end{aligned} \tag{15}$$

Thus, projective rational Bézier curves from Eq. (14) can be interpreted in terms of barycentres of their control points, weighted with the Bernstein basis and the associated weights. Returning to Cartesian coordinates consists in dividing the point by its homogeneous coordinate from Eq. (14), which corresponds to a projection onto the hyperplane $w = 1$ and leads to

$$\left(\sum_{i=0}^{d} B_i^d(t)\, w_i P_i,\ \sum_{i=0}^{d} B_i^d(t)\, w_i \right)^\top \doteq \left(\frac{\sum_{i=0}^{d} B_i^d(t)\, w_i P_i}{\sum_{i=0}^{d} B_i^d(t)\, w_i}, 1 \right)^\top \tag{16}$$

which exactly recovers the classical rational Bézier curve definition in Eq. (12) in homogeneous form, depicted in Fig. 2.

Note that this normalization is actually optional: it is only necessary when explicit Euclidean coordinates are required (e.g., for rendering the curve in Euclidean space). Otherwise, all projective computations and transformations can be performed directly in projective form, which preserves generality and avoids unnecessary divisions.

4 BEZIER CURVES WITHIN THE PGA FRAMEWORK

In the previous section, we introduced classical rational Bézier curves and their projective interpretation. In this section, we revisit rational Bézier curves through the lens of Projective Geometric Algebra (PGA).

4.1 Projective Bézier Curves in PGA

For the sake of generality, we consider n-dimensional (nd) plane-based PGA, where points are built from Eq. (6). Let $\hat{\mathbf{P}}_i$ denote a finite normalized point, as defined in Eq. (5), the projective rational Bézier curve can then be expressed as the following

$$\mathbf{P}(t) = \sum_{i=0}^{d} B_i^d(t)\, \lambda_i \hat{\mathbf{P}}_i \tag{17}$$

where, for now, $\lambda_i > 0$ denote the weights associated to $\hat{\mathbf{P}}_i$. The finite point norm defined in Eq. (7) requires that all $\hat{\mathbf{P}}_i$ share the same sign of their homogeneous coordinate, so that the weighted sum $\sum_{i=0}^{d} \lambda_i \hat{\mathbf{P}}_i$ defines a consistent barycentric combination, that is, either, the homogeneous coordinate, $w = +1$ for all i, or $w = -1$ for all i. We write λ_i for the weights to distinguish them from the homogeneous coordinate w within the multivector representation. Note that this expression is formally equivalent to the projective rational Bézier curve defined in Eq. (14), the essential difference is that the control points now belong to PGA.

Similarly to Eq. (16), the curve is projected back to Euclidean space by normalizing the computed homogeneous point. Thanks to the PGA metric, this corresponds to

$$\hat{\mathbf{P}}(t) = \frac{\mathbf{P}(t)}{\|\mathbf{P}(t)\|} \tag{18}$$

where, for any finite point $\mathbf{P}(t)$

$$\|\mathbf{P}(t)\| = w(t) = \left| \sum_{i=0}^{d} B_i^d(t)\, \lambda_i \right| \tag{19}$$

In practice, Eq. (19) does not need to be calculated, since the homogeneous coordinate of a finite point corresponds to its norm, which is already provided within the Eq. (17).

When all the $\lambda_i > 0$, the absolute value simplifies to $\sum_{i=0}^{d} B_i^d(t)\, \lambda_i$, which recovers exactly the rational Bézier curve in Eq. (12).

These formulations are particularly well suited to PGA: the weights are independent of the multivector representation of points, normalization remains optional, and all operations can be performed directly in projective space. When normalization is desired, the curve is simply divided by its norm in a coordinate-free manner, an intuitive and standard approach in geometric algebra that naturally follows from the metric structure.

In what follows, Eq. (17) and Eq. (18) serve as the foundational building blocks for generalizing Bézier curves in PGA.

4.2 Extend to Negative Weights

In the classical treatment of rational Bézier curves, weights are typically assumed to be strictly positive, which is why the previous subsection restricts them to $\lambda_i > 0$. This assumption guarantees several desirable geometric properties, most notably the convex hull property, and ensures that the denominator in the normalization step, Eq. (18), never vanishes, $\sum_{i=0}^{d} B_i^d(t)\lambda_i \neq 0 \quad \forall t \in [0,1]$.

As a result, the curve is well-defined as a Euclidean point for every parameter value t, and no singularities occur. Moreover, positive weights provide intuitive shape control: each control point acts attractively. This behavior is geometrically predictable and numerically stable, see [Far02].

However, allowing weights to take negative values modifies these properties and gives rise to a variety of different geometric phenomena. Garnier et al. address this in [GB16] without relying on the projective space. Their definition, together with its limitations, is discussed in detail in the following.

Within the PGA framework, singularities arise naturally and admit a consistent geometric interpretation. When negative weights are introduced, the point $\mathbf{P}(t)$ defined in Eq. (17) may no longer be a finite point, since for some $t \in [0,1]$

$$w(t) = \|\mathbf{P}(t)\|_f = \left| \sum_{i=0}^{d} B_i^d(t)\,\lambda_i \right|, \tag{20}$$

may vanish, in which case $\mathbf{P}(t)$ is an ideal-point. In this case, the general norm of Eq. (4) automatically switches to the ideal norm $\|\mathbf{P}(t)\|_\infty$, which remains nonzero, so that normalization is still well-defined. Geometrically, this indicates that the curve passes through an ideal-point, manifesting as an asymptotic direction encoded directly in $\mathbf{P}(t)$.

Introducing negative weights therefore provides a natural geometric interpretation. While positive weights attract the curve toward the corresponding control point, a negative weight reverses the sign of its contribution. The resulting effect often appears as a repulsive influence, as illustrated in Fig. 3.

4.2.1 *Complementary curve section*

Changing the sign of the endpoint weights produces a simple geometric transformation of a rational Bézier curve that can be interpreted using the tangent directions at the endpoints. Consider a rational Bézier curve $\hat{\mathbf{P}}(t)$ of degree d defined in Eq. (18), the tangent directions at the endpoints are well known and depend only on the ratios of adjacent weights. They are given by the tangent directions $\mathbf{P}'(0)$ at $t = 0$ and $\mathbf{P}'(1)$ at $t = 1$, as follows

$$\mathbf{P}'(0) \propto \frac{\lambda_1}{\lambda_0}(\mathbf{P}_1 - \mathbf{P}_0), \qquad \mathbf{P}'(1) \propto \frac{\lambda_{d-1}}{\lambda_d}(\mathbf{P}_d - \mathbf{P}_{d-1}). \tag{21}$$

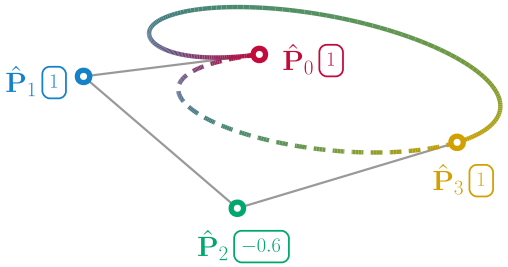

Figure 3: Cubic rational Bézier curves illustrating the influence of negative weights. The solid curve corresponds to the control point P_2 with $\lambda = -0.6$, while the dashed curve corresponds to P_2 with $\lambda = 0.6$.

Now consider the transformation

$$\lambda_0 \to -\lambda_0, \qquad \lambda_d \to -\lambda_d, \tag{22}$$

while all other weights remain unchanged. Let $\mathbf{P}_c(t)$ denote the Bézier curve obtained after changing the sign of the endpoint weights. The endpoint tangents become

$$\mathbf{P}_c'(0) = -\mathbf{P}'(0), \qquad \mathbf{P}_c'(1) = -\mathbf{P}'(1). \tag{23}$$

Therefore the tangent directions at both endpoints are reversed. The curve still interpolates the same endpoints $\mathbf{P}_0$ and $\mathbf{P}_d$, but it leaves them in the opposite directions, conserving the tangent norms along the same tangent lines.

Geometrically, the tangents at $\mathbf{P}_0$ and $\mathbf{P}_d$ partition the supporting curve into two possible curve sections connecting the endpoints. Reversing the tangent directions forces the curve to follow the other curve section determined by these tangent constraints. Consequently, the new rational Bézier curve traces the complementary section of the same underlying algebraic curve. This phenomenon is illustrated in Fig. 4.

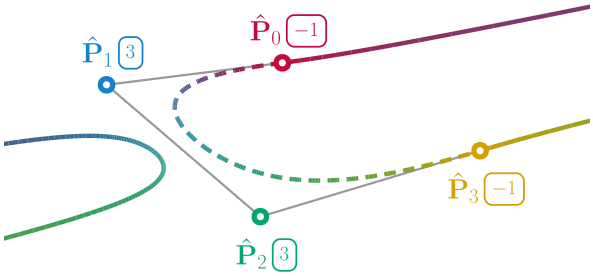

Figure 4: Complementary cubic rational Bézier curves obtained by negating the endpoint weights. The solid curve corresponds to the displayed weights, the dashed curve corresponds to the following weight inversion $(\lambda_0 \to -\lambda_0, \lambda_3 \to -\lambda_3)$.

4.2.2 *Conic Sections Case Study*

As mentioned before, negative weights in rational Bézier curves play an important role in representing

curve segments that extend to infinity. A classical example illustrating this behavior is the quadratic case, which naturally represents conic sections.

The Bernstein basis functions used in a quadratic Bézier curve for $t \in [0,1]$ are

$$B_0(t) = (1-t)^2, \qquad B_1(t) = 2t(1-t), \qquad B_2(t) = t^2. \tag{24}$$

The corresponding quadratic rational Bézier curve is

$$\hat{\mathbf{P}}(t) = \frac{(1-t)^2\lambda_0\hat{\mathbf{P}}_0 + 2t(1-t)\lambda_1\hat{\mathbf{P}}_1 + t^2\lambda_2\hat{\mathbf{P}}_2}{(1-t)^2\lambda_0 + 2t(1-t)\lambda_1 + t^2\lambda_2}. \tag{25}$$

The denominator polynomial plays a central role in determining the projective behavior of the curve. For the quadratic case, it expands as

$$\begin{aligned} Q(t) &= (1-t)^2\lambda_0 + 2t(1-t)\lambda_1 + t^2\lambda_2 \\ &= \lambda_0 + 2(-\lambda_0+\lambda_1)t + (\lambda_0 - 2\lambda_1 + \lambda_2)t^2. \end{aligned} \tag{26}$$

The roots of $Q(t)$ correspond to parameter values for which the homogeneous point reaches the ideal line. Hence the existence of real roots determines whether the curve intersects the line at infinity. This is governed by the discriminant

$$\Delta = 4(\lambda_1^2 - \lambda_0\lambda_2). \tag{27}$$

Therefore the sign of the invariant quantity Δ determines the projective type of the conic:

- if $\Delta < 0$, the denominator never vanishes for real t. The conic is of elliptic type and does not intersect the ideal line.
- if $\Delta = 0$, the denominator has a double root. The conic is parabolic and is tangent to the ideal line.
- if $\Delta > 0$, two distinct real roots exist. The conic is hyperbolic and intersects the ideal line in two points.

The quantity appearing in Eq. (27) is invariant under the sign inversions $\lambda_0 \to \lambda_0$ and $\lambda_2 \to -\lambda_2$ as well as $\lambda_1 \to -\lambda_1$. Consequently, these transformations do not change the projective type of the conic. However, they do modify the parametrized portion of the conic traced by the Bézier curve, as detailed in Sec. 4.2.1 above. Examples of this complementary construction for conic curve segments are illustrated in Fig. 5.

4.3 Finite and Ideal Points

In the previous subsection, we extended weights to take not only positive but also negative values. In the classical Bézier curve formulation, control points with zero weight correspond to vectors. These elements allow the curve to represent asymptotic directions, tangents, or other directional behaviors. As discussed earlier, Garnier et al. [GB16] study rational Bézier curves through the concept of mass point control objects. Their mass point formalism permits control elements to have both

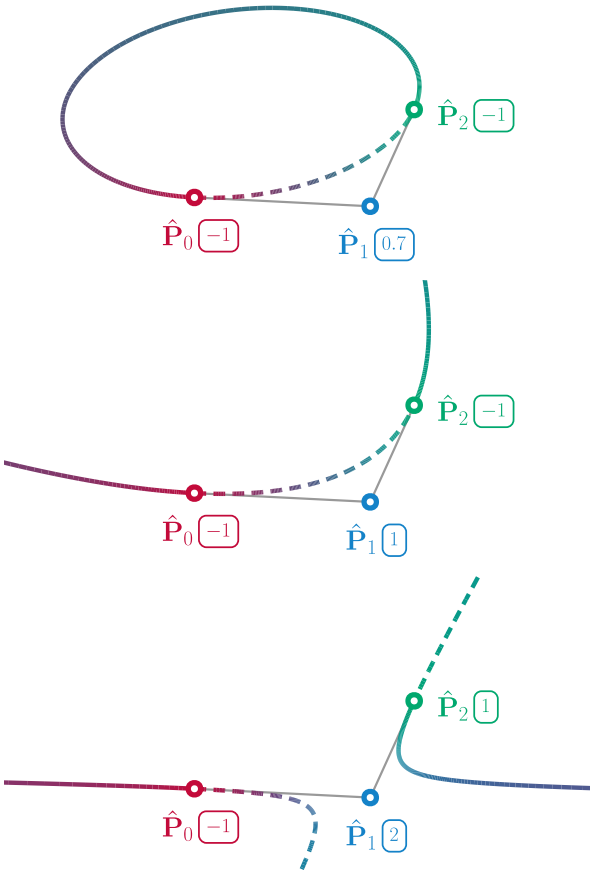

Figure 5: Conics constructed as complementary rational Bézier curves. Solid curves correspond to the represented endpoint weights, dashed curves correspond to complementary sections. The three cases shown are an ellipse, a parabola, and a hyperbola.

negative and zero weights. Within this framework, finite points P_i with nonzero weights w_i are distinguished from zero-weight vectors V_i, leading to a curve representation of the form

$$P(t) = \frac{\overbrace{\sum_{i\in I} B_i^d(t)\, w_i P_i}^{\text{points}} + \overbrace{\sum_{i\in J} B_i^d(t)\, V_i}^{\text{vectors}}}{\sum_{i\in I} B_i^d(t)\, w_i} \tag{28}$$

where $P_i, V_i \in \mathbb{R}^2$, $w_i \in \mathbb{R}$, and the index sets defined as $I = \{i \in [0,d] \mid w_i \neq 0\}$ and $J = \bar{I}$ partition $\{0,\ldots,d\}$ into finite-point and vector control elements respectively.

While this separation is meaningful in Euclidean space, it does not generalize naturally to projective coordinates. In particular, it prevents assigning independent weights to vectors: in their formalism, the weight associated with a vector is implicitly encoded in its magnitude, rather than being an explicit degree of freedom.

In PGA, finite and ideal points are treated uniformly, so the projective curve is still given by Eq. (17), with $\hat{\mathbf{P}}_i$

now denoting either a normalized finite or ideal point, each carrying an independent weight λ_i. Normalization is retained via Eq. (18). Since ideal points have a vanishing homogeneous coordinate, they do not contribute to $\|\mathbf{P}(t)\|$ with $\mathbf{P}(t)$ a finite point, so that only the weights of finite control elements appear in the normalization sum. In practice, the algebraic structure of Eq. (18) already accounts for the correct norm.

Note that we further restrict to $\lambda_i \neq 0$ for all i. Indeed, a control element with weight $\lambda_i = 0$ would cancel the corresponding control point contribution, breaking the partition of unity of the weighted Bernstein basis. The resulting curve would no longer be a well-defined rational Bézier curve of degree d in the classical sense: it would retain the degree-d Bernstein polynomials while having fewer than $d+1$ active control elements, placing it outside the standard rational Bézier framework.

If $\hat{\mathbf{P}}_0$ or $\hat{\mathbf{P}}_d$ is an ideal-point, the endpoint interpolation mechanism is preserved: the curve interpolates the ideal-point at $t = 0$ or $t = 1$, meaning it arrives from or departs toward infinity along the asymptotic direction encoded by the ideal-point, as illustrated in Fig. 6.

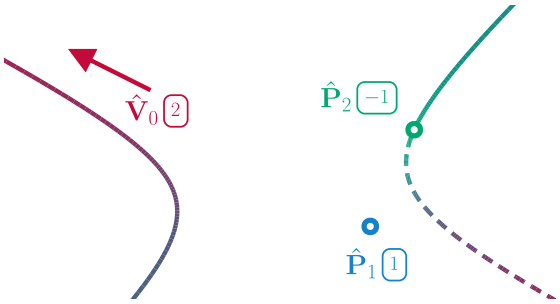

Figure 6: Cubic rational Bézier curve with one ideal point at an endpoint. The solid curve corresponds to the displayed weights, the dashed curve shows the complementary section obtained by negating the endpoint weights.

If an interior control element $\hat{\mathbf{P}}_k$ with $0 < k < d$ is an ideal-point, it acts as a directional contribution on the curve formation: for parameter values where $B_k^d(t)$ is significant, the curve is attracted toward the direction encoded by $\hat{\mathbf{P}}_k$, without being anchored to any finite location. The weight λ_k controls the intensity of this directional influence, with $\lambda_k < 0$ reversing the direction of attraction. An example is depicted in Fig. 7.

4.3.1 *Complementary curve section*

As established in Sec. 4.2.1, negating the endpoint weights $\lambda_0 \to -\lambda_0$ and $\lambda_d \to -\lambda_d$ reverses both endpoint tangent directions and selects the complementary arc of the same conic, regardless of whether $\hat{\mathbf{P}}_1$ is a finite or ideal-point, as illustrated in Fig. 7 and Fig. 6.

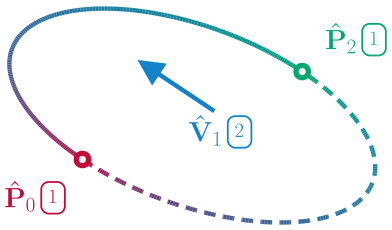

Figure 7: Cubic rational Bézier curve defined by three finite control points and one transition ideal point. The solid curve corresponds to the displayed weights, the dashed curve shows the complementary section obtained by negating the endpoint weights

4.3.2 *Conic Sections Case Study*

To analyse the effect of ideal control points on the projective type of the conic, we return to the denominator polynomial defined in Eq. (26), whose discriminant governs whether the curve reaches the ideal line. As established in Sec. 4.2.2, the general discriminant is $\Delta = 4(\lambda_1^2 - \lambda_0\lambda_2)$.

When one of the control points is an ideal-point, its associated weight does not contribute to the denominator of Eq. (25) and the discriminant takes a degenerate form. Two distinct cases arise naturally. If $\hat{\mathbf{P}}_1$ is an ideal-point, the effective discriminant then reduces to $\Delta_{\lambda_1} = -4\lambda_0\lambda_2$, so the projective type of the conic is determined only by the two finite endpoint weights.

Whenever both endpoint weights share the same sign, $\lambda_0\lambda_2 > 0$ then $\Delta_{\lambda_1} < 0$, giving an elliptic conic (Fig. 7). Conversely, $\lambda_0\lambda_2 < 0$ yields $\Delta_{\lambda_1} > 0$, corresponding to a hyperbola. The parabolic case $\Delta_{\lambda_1} = 0$ is degenerate and requires one of the endpoint weights to vanish.

If instead, one of the endpoint control points is ideal, its homogeneous weight coordinate vanishes, and the corresponding weight λ_0 or λ_2 therefore does not appear in the denominator polynomial. Taking $\lambda_0 = 0$ or $\lambda_2 = 0$, the discriminant reduces to $\Delta_{\lambda_0} = \Delta_{\lambda_2} = 4\lambda_1^2$.

Since $\lambda_1^2 \geq 0$ with equality only in the degenerate case $\lambda_1 = 0$, one has $\Delta \geq 0$ throughout. The conic is therefore always of hyperbolic (Fig. 6) or parabolic type: hyperbolic for λ_1 associated to a finite point, and parabolic in the degenerate case, for λ_1 associated to an ideal-point.

4.4 Ideal-Points Bézier Curves

The previous subsection introduced the use of ideal points to describe directional influence within the finite curve. In the limiting case where all control elements are ideal points, the curve no longer traces a locus of positions in Euclidean space. Instead, it defines a smoothly varying family of directions, a Bézier-based interpolation between directions rather than between

positions. In this limiting case, $\mathbf{P}(t)$ remains an ideal point for all $t \in [0,1]$, and normalization via $\|\mathbf{P}(t)\|_\infty$ remains well-defined, yielding a smoothly varying unit direction $\hat{\mathbf{P}}(t)$.

Geometrically, this construction defines a rational curve on the projective sphere of directions, interpolating between the asymptotic directions $\hat{\mathbf{P}}_0$ and $\hat{\mathbf{P}}_d$ at $t = 0$ and $t = 1$ respectively. The intermediate control directions $\hat{\mathbf{P}}_1, \ldots, \hat{\mathbf{P}}_{d-1}$, shape the angular path and the weights λ_i modulate the rate of angular progression along it, analogously to how weights influence the curve shape in the finite-point case. An illustration of an ideal-point Bézier curve is depicted in Fig. 8.

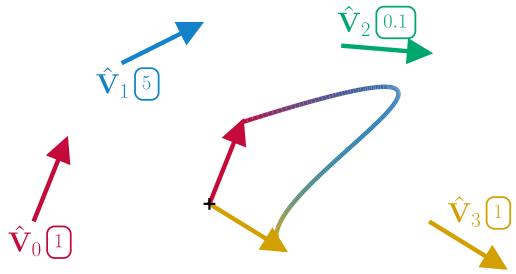

Figure 8: Cubic (non-rational) Bézier curve with all-ideal control points, defining a smooth interpolation of directions rather than positions.

This demonstrates that the PGA formulation of rational Bézier curves is truly unified: the same equations, Eq. (17) and Eq. (18), cover finite curves, curves with mixed finite and ideal control points, and purely directional curves, all distinguished only by the nature of their control elements and handled consistently by the metric structure of PGA. This unified representation simplifies computations, avoids exceptional cases, and naturally extends rational Bézier curves to configurations involving both finite and ideal-points.

5 GENERAL PGA BEZIER CURVE

In the previous sections, we reformulated Bézier curves in the framework of Projective Geometric Algebra (PGA) using finite and ideal points as control elements. One of the central strengths of geometric algebra is that it provides a unified representation of geometric entities of different types within the same algebraic structure. In this section, we extend the Bézier formalism beyond points to arbitrary geometric objects and ultimately to transformation objects represented as versors.

The key observation is that elements of a fixed grade form a linear space: they can be freely combined through linear combinations without leaving that space. Since the Bernstein basis forms a partition of unity, any weighted combination of grade-k elements remains a grade-k element, and thus a geometric object of the same type. This is precisely what allows the rational Bézier construction to generalize naturally from curves of points to curves of arbitrary geometric elements, while preserving the same blending structure.

5.1 Geometric Objects Bézier Curves

As detailed in Sec. 2, geometric objects in PGA can be constructed as intersections of hyperplanes (or as joins of lower-dimensional objects, see [DFM07] for further details) or directly from their implicit forms as in Eq. (1).

In 2D PGA, hyperplanes reduce to lines, constructible either as the join of two points or directly from their implicit form

$$\mathbf{L} = a\,\mathbf{e}_1 + b\,\mathbf{e}_2 + c\,\mathbf{e}_0. \tag{29}$$

where a, b and c correspond exactly to the Hessian form of a line $ax+by+cw=0$. In PGA, the implicit equation can be rewritten as $\mathbf{L}^\star \cdot \mathbf{P} = 0$ if $\mathbf{P}$ lies on $\mathbf{L}$.

Substituting line elements for point control elements in Eq. (17) yields a smoothly blended family of lines

$$\mathbf{L}(t) = \sum_{i=0}^{d} B_i^d(t)\,\lambda_i\,\hat{\mathbf{L}}_i, \tag{30}$$

where $\hat{\mathbf{L}}_i$ are the normalized control line elements and $\lambda_i \neq 0$ their associated weights, in direct analogy with the point case. An example is depicted in Fig. 9.

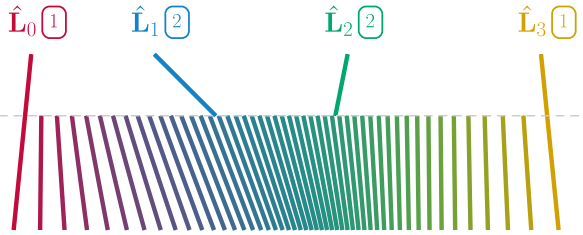

Figure 9: Cubic Bézier curve of lines in 2d PGA. The control lines are shown at the top, and the resulting Bézier curve of lines is depicted at the bottom.

In 3D PGA, hyperplanes reduce to planes, constructible either as the join of three points or from their implicit form defined in Eq. (1). The construction carries over identically, yielding a smoothly blended family of planes:

$$\mathbf{\Pi}(t) = \sum_{i=0}^{d} B_i^d(t)\,\lambda_i\,\hat{\mathbf{\Pi}}_i, \tag{31}$$

where $\hat{\mathbf{\Pi}}_i$ are the normalized control plane elements and $\lambda_i \neq 0$ their associated weights. An example is depicted in Fig. 10.

More generally, in n-dimensional PGA, the same construction extends to hyperplanes, yielding a Bézier curve of hyperplanes

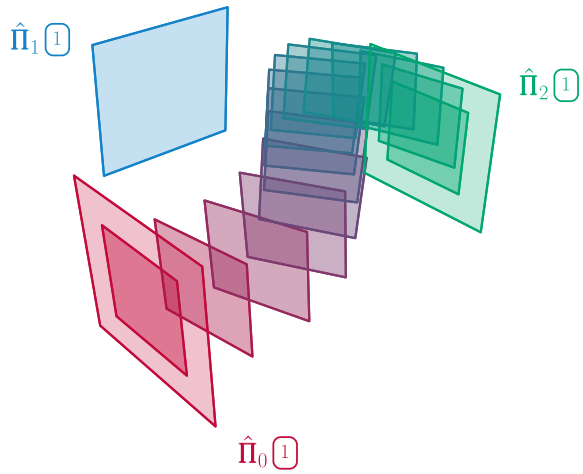

Figure 10: Rational Bézier curve of planes in 3d PGA. The control planes are represented as large squares, while the resulting curve of planes is depicted with smaller squares.

$$\mathbf{h}(t) = \sum_{i=0}^{d} B_i^d(t)\,\lambda_i\,\hat{\mathbf{h}}_i. \tag{32}$$

Note that ideal elements are naturally included in this construction: in the 2D case, the ideal hyperplane is the line at infinity $\mathbf{L}_\infty = \mathbf{e}_0$; in the 3D case, it is the plane at infinity $\mathbf{\Pi}_\infty = \mathbf{e}_0$. As in the point case, normalization via Eq. (18) remains available whenever unit geometric objects are required, with the adaptive norm of Eq. (4) handling finite and ideal elements uniformly.

More generally, these constructions are available for all geometric objects of the algebra, including 3d lines.

5.2 Versor Bézier Curves

The previous subsection extended the Bézier formalism to all finite and ideal geometric objects of PGA. As discussed in Sec. 2, PGA also represents geometric transformations such as rotations, translations, and reflections uniformly as versors acting on any object of the algebra via the sandwich product. Since versors are represented as multivectors within the algebra, their linear combinations are well-defined. The resulting blend $V(t)$ is a multivector that can be renormalized at each t via Eq. (5) to recover a unit versor. A Bézier curve of versors is therefore defined as

$$V(t) = \sum_{i=0}^{d} B_i^d(t)\,\hat{V}_i\lambda_i, \qquad \hat{V}(t) = \frac{V(t)}{\|V(t)\|}, \tag{33}$$

where $\hat{V}_i$ are the control unit versors and $\lambda_i \neq 0$ their associated weights.

Motors are specific versors of even grade that encode rigid body motions, combining rotations and translations within a single multivector, in direct correspondence with dual quaternions. For example in 2D PGA, a motor can be defined as

$$M = \exp\left(\tfrac{1}{2}\mathbf{P}\right), \tag{34}$$

where $\mathbf{P}$ is a point encoding the axis and magnitude of the motion. This is the PGA analogue of the quaternion exponential used in classical rotation interpolation. A Bézier curve of motors is obtained by substituting motor control elements into the general versor construction

$$M(t) = \sum_{i=0}^{d} B_i^d(t)\,\hat{M}_i\lambda_i, \qquad \hat{M}(t) = \frac{M(t)}{\|M(t)\|}. \tag{35}$$

Since motors belong to the even subalgebra, see [DFM07], their linear combinations remain well-defined multivectors. However, the blend $M(t)$ is not guaranteed to satisfy $\|M(t)\| = 1$, so renormalization is required.

The normalized curve $\hat{M}(t)$ interpolates the endpoint motors M_0 and M_d at $t = 0$ and $t = 1$ respectively, and defines a smooth one-parameter family of rigid motions. At each parameter value t, the transformation $\hat{M}(t)$ can be applied to any PGA object via the sandwich product of Eq. (9), providing a direct and unified mechanism for animating geometric objects along a smooth motion path. An example of the effect of this kind of curve is depicted in Fig. 11.

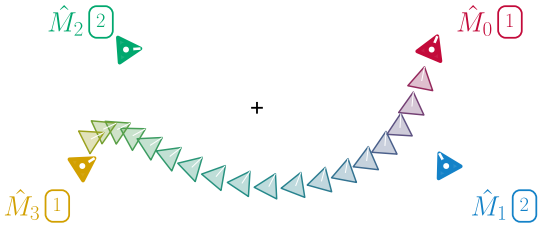

Figure 11: Cubic Bézier curve of motors. The control motors are illustrated by the position and orientation of filled triangles, while the resulting curve of motors is depicted with hollow triangles.

It should be noted that this linear blending approach, while simple and efficient, does not follow geodesics on the motor Lie group. For applications requiring geodesic interpolation, such as constant-speed rigid body motion, the exponential and logarithmic maps should be used instead, as in the classical Screw Linear Interpolation (ScLERP) [DKR22]. Extending the proposed framework to incorporate such maps remains an open direction for future work.

6 IMPLEMENTATION

All constructions presented in this paper were implemented using the `kingdon` library [Roe25], a Python package for symbolic and numeric computation in geometric algebra, with which all figures were produced. To let readers reproduce and explore each construction directly, the same code is packaged as a set of interactive Jupyter notebooks deployed as a static site[1] that runs entirely in the browser. For a computationally efficient implementation, the reader can refer to `klein`[2].

7 CONCLUSION

This paper presented a reformulation of Bézier curves within Projective Geometric Algebra (PGA), unifying rational Bézier curves, negative weights, ideal control points, and Bézier interpolation of geometric objects and versors under a single algebraic structure, demonstrating the expressive power of PGA for curve modeling. Several directions remain open for future work. Extending the framework to Bézier surfaces and spline constructions within PGA would be a natural continuation. The versor interpolation presented here relies on linear blending, which does not follow geodesics on the motor Lie group; incorporating exponential and logarithmic maps to achieve geodesic interpolation represents an important avenue for applications requiring constant-speed rigid body motion. Finally, investigating the computational efficiency of these constructions within optimized geometric algebra libraries and their integration into existing CAD pipelines remains a promising direction.

8 REFERENCES

[BDHBC23] Johann Brehmer, Pim De Haan, Sönke Behrends, and Taco S Cohen. Geometric algebra transformer. *Advances in Neural Information Processing Systems*, 36:35472–35496, 2023.

[BÉZ92] Pierre BÉZIER. *Courbes et surfaces pour la CFAO*. Ed. Techniques Ingénieur, 1992.

[dB78] Carl de Boor. *A Practical Guide to Spline*, volume Volume 27. 01 1978.

[DC85] Paul De Casteljau. Mathématiques et cao. volume 2: formes à pôles. 1985.

[DDK22] Leo Dorst and Steven De Keninck. A guided tour to the plane-based geometric algebra pga. *URL https://bivector.net/PGA4CS.html*, 2022.

[DFM07] Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric Algebra for Computer Science, An Object-Oriented Approach to Geometry*. Morgan Kaufmann, 2007.

[DK20] Steven De Keninck. ganja.js, 2020.

[DKR22] Steven De Keninck and Martin Roelfs. Normalization, square roots, and the exponential and logarithmic maps in geometric algebras of less than 6d. *Mathematical Methods in the Applied Sciences*, 47(3):1425–1441, August 2022.

[DL03] Chris Doran and Anthony Lasenby. *Geometric Algebra for Physicists*. Cambridge University Press, 2003.

[Far02] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Morgan Kaufmann, San Francisco, CA, 5th edition, 2002.

[FH00] Gerald Farin and Dianne Hansford. *The essentials of CAGD*. AK Peters/CRC Press, 2000.

[GB16] Lionel Garnier and Jean-Paul Bécar. Mass points, bézier curves and conics: a survey. In *Eleventh International Workshop on Automated Deduction in Geometry, Proceedings of ADG*, pages 97–116, 2016.

[Gun17] Charles G Gunn. Doing euclidean plane geometry using projective geometric algebra. *Advances in Applied Clifford Algebras*, 27(2):1203–1232, 2017.

[Had23] Hugo Hadfield. *Applications of Geometric Algebra in Mathematical Engineering*. PhD thesis, University of Cambridge (United Kingdom), 2023.

[Per09] Christian Perwass. *Geometric algebra with applications in engineering*, volume 4 of *Geometry and Computing*. Springer, 2009.

[PT12] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[RGDK+23] David Ruhe, Jayesh K Gupta, Steven De Keninck, Max Welling, and Johannes Brandstetter. Geometric clifford algebra networks. In *International Conference on Machine Learning*, pages 29306–29337. PMLR, 2023.

[Roe25] Martin Roelfs. The willing kingdon clifford algebra library, 2025.

[Som13] Gerald Sommer. *Geometric computing with Clifford algebras: theoretical foundations and applications in computer vision and robotics*. Springer Science & Business Media, 2013.

[1] `https://eharquin.github.io/pga-bezier/`
[2] `https://github.com/jeremyong/klein/`