Computer Aided Geometric Design 43 (2016) 16–26

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Revisiting the problem of zeros of univariate scalar Béziers

Jinesh Machchhar [*], Gershon Elber

Faculty of Computer Science, Technion Israel Institute of Technology, Israel

ARTICLE INFO

Article history:
Available online 18 February 2016

Keywords:
Polynomial roots
Bézier polynomials
Subdivision
Numerical method
Newton–Raphson, polynomial division

ABSTRACT

This paper proposes a fast algorithm for computing the real roots of univariate polynomials given in the Bernstein basis. Traditionally, the polynomial is subdivided until a root can be isolated. In contrast, herein we aim to find a root only to subdivide the polynomial at the root. This subdivision based algorithm exploits the property that the Bézier curves interpolate the end-points of their control polygons. Upon subdivision at the root, both resulting curves contain the root at one of their end-points, and hence contain a vanishing coefficient that is factored out. The algorithm then recurses on the new sub-curves, now of lower degree, yielding a computational efficiency. In addition, the proposed algorithm has the ability to efficiently count the multiplicities of the roots. Comparison of running times against the state-of-the-art on thousands of polynomials shows an improvement of about an order-of-magnitude.

1. Introduction and previous work

The problem of numerically finding zeros of univariate polynomials is ubiquitous in computer aided design (Cohen et al., 2001) and engineering. Many geometric problems can be cast into that of finding zeros of polynomials, for instance, computing intersections of curves and surfaces (Sederberg and Meyers, 1988; Sederberg and Parry, 1986), contact analysis of shapes (Kim et al., 2014), kinematic analysis (Bartoň et al., 2009), etc. There have been many approaches to the problem of computing zeros of univariate polynomials in the past (Isaacson and Keller, 1966; McNamee, 1990), such as, based on Newton's method (Grandine, 1989), using Descartes' rule of signs (Eigenwillig et al., 2006; Krandick and Mehlhorn, 2006; Rouillier and Zimmermann, 2004), based on subdivision (Gopalsamy et al., 1991; Mourrain and Pavone, 2009), to name a few.

In this work, we use the Bernstein representation for polynomials. Bernstein polynomials have several useful properties such as the variation diminishing property, the convex hull property and numerical stability with respect to perturbation of coefficients (Farouki and Rajan, 1987), which makes such a representation especially amenable for numerical applications.

One of the earliest methods to exploit the variation diminishing property of Bézier curves in order to isolate the roots of polynomials was given by Lane and Riesenfeld (1981). In 1990, Sederberg and Nishita (1990) proposed the technique of Bézier clipping for identifying regions of the domain which contain roots. This was done by intersecting the convex hull of the control polygon with the zero axis. In 2007, Bartoň and Jüttler (2007) improved the technique of Bézier clipping using degree reduction to generate a strip bounded by two quadratic polynomials, which encloses the graph of the input polynomial. This strip, when intersected with the zero axis, gives the new interval potentially containing the roots. This approach, that is also known as *quadratic clipping*, was shown to have quadratic convergence by Schulz (2009). In 2009,

[*] Corresponding author.
E-mail addresses: jineshmac@cs.technion.ac.il (J. Machchhar), gershon@cs.technion.ac.il (G. Elber).

http://dx.doi.org/10.1016/j.cagd.2016.02.017
0167-8396/© 2016 Elsevier B.V. All rights reserved.

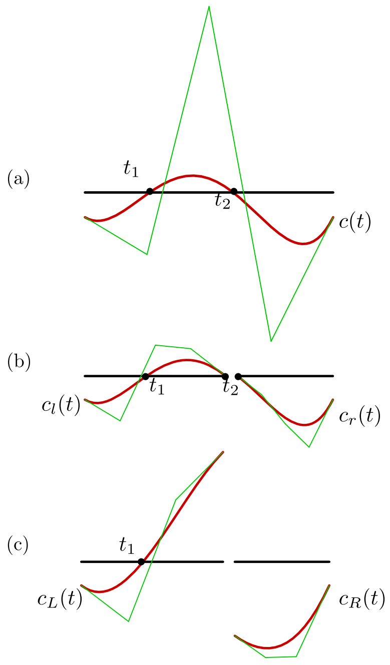

Fig. 1. Overview of our approach. (a) Shows the graph of a degree 4 Bézier polynomial $c(t)$ in red with roots at t_1 and t_2, along with its control polygon in green. The t-axis is shown in black. (b) Shows the polynomials $c_l(t)$ and $c_r(t)$ obtained after subdividing $c(t)$ at t_2. $c_l(t)$ and $c_r(t)$ are again of degree 4 and contain the root at t_2. (c) shows the degree 3 polynomials obtained after eliminating the root at t_2 from $c_l(t)$ and $c_r(t)$, factoring out $(1-t)$ and t, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Liu et al. (2009) improved quadratic clipping by using cubic polynomials, yielding faster rates of convergence. In 2007, Mørken and Reimers (2007) utilized the close relationship between the spline and its control polygon for computing zeros of polynomials. They use the zeros of the control polygon as an initial guess for tracing the zeros of the polynomial. The control polygon is iteratively refined until the roots are found. In 2013, Ko and Kim (2013) used bounding polygons to reduce the intervals containing roots of Bézier polynomials. A hybrid of the convex hull, sharp bounds (Nairn et al., 2006) and quasi-interpolating bounds (Zhang and Wang, 2006) was used to refine the intervals. Recently, in 2015, Chen et al. (2015) improved the convergence rates achieved by Liu et al. (2009) by bounding the polynomial of interest by a pair of rational cubic polynomials.

Our approach, as explained in Section 1.1 uses the fact that polynomials represented in the Bernstein–Bézier form admit efficient algorithms for polynomial multiplication and division (Buse and Goldman, 2007; Goldman, 2002). These are employed to factor out the roots already computed, thus reducing the degree of the polynomial, as part of the solution process.

1.1. Overview of our approach

We now give an overview of our method, which exploits several properties of the Bézier curves, for computing zeros of scalar polynomials. The Bernstein basis for an n-degree univariate polynomial is given by the set of functions $\theta_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$, $t \in [0,1]$, for $i = 0, \ldots, n$. A degree n scalar polynomial in the Bernstein basis is expressed as $c(t) = \sum_{i=0}^{n} p_i \theta_{i,n}(t)$ where, $p_i \in \mathbb{R}$ are the real coefficients of the polynomial. While $c(t)$ may have roots outside the domain $[0,1]$, in this paper we focus on finding the real roots of $c(t)$ in $[0,1]$. Throughout this paper, we will consider $[0,1]$ to be the domain of definition of all Bézier curves, unless stated otherwise.

We will exploit the fact that the Bézier curves interpolate the end-points of their control polygon. Upon finding a root, t_0, our algorithm subdivides $c(t)$ at t_0. Let $c_l(t)$ and $c_r(t)$ denote the resulting polynomials corresponding to the sub-intervals $[0,t_0]$ and $[t_0,1]$, respectively, with their domains again mapped to $[0,1]$. Clearly, $c_l(t)$ and $c_r(t)$ vanish at 1 and 0, respectively. Since the Bézier curves interpolate the end-points of their control polygon, it follows that the last coefficient of $c_l(t)$ and the first coefficient of $c_r(t)$ are zero. $c_l(t)$ may be expressed as $(1-t)c_L(t)$ for some Bernstein polynomial $c_L(t)$ having one degree less than $c_l(t)$. Similarly, $c_r(t)$ may be written as $tc_R(t)$ for some Bernstein polynomial $c_R(t)$ with one degree less than $c_r(t)$. Factoring out the terms $(1-t)$ and t from $c_l(t)$ and $c_r(t)$, respectively (Buse and Goldman, 2007), eliminates the root t_0 from these two polynomials, yielding polynomials with smaller degrees to recurse upon that preserves all roots but t_0, thus giving a computational boost to the algorithm. This is explained schematically in Fig. 1. The graph of polynomial $c(t)$ is shown in red, in Fig. 1(a), along with its control polygon that is shown in green. $c(t)$ has two real roots, at t_1 and t_2. Assume we found the root at t_2. Fig. 1(b) shows the subdivided polynomials $c_l(t)$ and $c_r(t)$ respectively, both containing