Computer-Aided Design 43 (2011) 1870–1878

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Solving polynomial systems using no-root elimination blending schemes

Michael Bartoň *

KAUST — King Abdullah University of Science and Technology, Geometric Modeling and Scientific Visualization Center, Thuwal, Saudi Arabia

ARTICLE INFO

Article history:
Received 30 April 2011
Accepted 19 September 2011

Keywords:
Polynomial systems
Linear blend
No-root criterion
Surface–surface–surface intersection

ABSTRACT

Searching for the roots of (piecewise) polynomial systems of equations is a crucial problem in computer-aided design (CAD), and an efficient solution is in strong demand. Subdivision solvers are frequently used to achieve this goal; however, the subdivision process is expensive, and a vast number of subdivisions is to be expected, especially for higher-dimensional systems. Two blending schemes that efficiently reveal domains that cannot contribute by any root, and therefore significantly reduce the number of subdivisions, are proposed. Using a simple linear blend of functions of the given polynomial system, a function is sought after to be no-root contributing, with all control points of its Bernstein–Bézier representation of the same sign. If such a function exists, the domain is purged away from the subdivision process. The applicability is demonstrated on several CAD benchmark problems, namely surface–surface–surface intersection (SSSI) and surface–curve intersection (SCI) problems, computation of the Hausdorff distance of two planar curves, or some kinematic-inspired tasks.

1. Introduction and previous work

Solving (piecewise) polynomial systems of equations is a crucial problem in many fields such as computer-aided design, manufacturing, robotics and kinematics. A robust and efficient solution is in strong demand. The symbolically oriented approaches such as *Gröbner bases* and similar elimination-based techniques [1] map the original system to a simpler one, preserving the solution set. In contrast to this, *polynomial continuation methods* start at roots of a suitable simple system and transform it continuously to the desired one [2]. These methods are very general and give global information about the solution set, regardless of the domain of interest. Typically, they operate in $\mathbb{C}^n$ and when only real solutions are sought; hence these methods can be inefficient.

The other approach is represented by a family of subdivision-based solvers, which typically treat the equations of the system as (parts of) hypersurfaces in $\mathbb{R}^n$, and search for its (real) intersection points inside some particular domain, usually a box in $\mathbb{R}^n$. The idea of a subdivision-based algorithm appeared in the early 1980s [3,4], when new algorithms for the evaluation of polynomial spline curves and surfaces were introduced and recursively applied on the CAD benchmark problems such as curve–curve and surface–surface intersections.

The *interval projected polyhedra* algorithm [5] employs Bernstein–Bézier representations of polynomials and projects its control points into two-dimensional (2D) subspaces where corresponding *convex hulls* are computed and intersected.

In order to reduce the number of computationally demanding subdivision steps or improve the robustness of the subdivision process, local preconditioning may be applied [6]. This technique is considered only for *well-constrained*, or also squared, $(n \times n)$ systems of equal-degree constraints. Various other methods for solving squared systems exist, and many related references can be found in [7].

For such systems, whose solution is, in general, a zero-variate set, a *termination criterion* is presented in [8]. This geometrically oriented scheme detects regions with *at most one* root and allows the application of techniques such as the multivariate Newton–Raphson method. Recently, the generalization for *non-polynomial* (transcendental) systems was introduced in [9]. Nevertheless, even if there is at most one root inside the domain, there is no guarantee that the Newton–Raphson method will converge to that root. One could apply a stricter criterion such as the Kantorovich condition [10] to avoid convergence failures in the numerical improvement stage.

The complexity of subdivision-based solvers is exponential in the dimension of the problem, when tensor product representations are used. In [11], expression trees are employed, reducing the expected complexity to polynomial.

Another family of subdivision-based solvers relies on *affine* or *interval arithmetic* [12]. These methods typically construct an interval bound on values that the given function may attain over a given domain. If the bound of some function of the system is a no-zero-containing bound, the particular domain is discarded. With these schemes, it is difficult to guarantee numerical stability during subdivision, and no root isolations are offered.

* Tel.: +966 2 808 0247.
E-mail address: michael.barton@centrum.cz.

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.09.011

Another approach is based on the reduction of the original n-dimensional system to a one-dimensional nonlinear equation; see [13] and related work cited therein. Every function of the system is evaluated at $n-1$ variables and solved with respect to the remaining one. The root of the univariate function and the partial derivatives of $f_i, i = 1, \ldots, n-1$ in the root are then used as the coefficients of the linear system which computes the improvement of the first $n-1$ values. The process is iteratively repeated, converging quadratically to the root.

Another iterative method was proposed in [14]. Every function of the system is considered as an *objective* function. The goal is to minimize these functions, and the problem is essentially reduced to a multi-objective optimization problem. An evolutionary algorithm is proposed and a sequence of candidates that approximate the root is created. Similarly to [13], the process is iterative, and a good initial guess is required to reach the root.

Recently, a general solver for overconstrained/well-constrained /underconstrained systems of equations, relying on the representation of polynomials in the *barycentric Bernstein basis* and exploiting the projecting control polyhedra algorithm, has been presented [15]. In the case of the zero-variate solution set, the sequence of n-dimensional root-containing bounding *simplices* is returned.

If the system is underconstrained by one equation, having n equations in $n+1$ variables, the solution is, in general, a curve in $\mathbb{R}^{n+1}$. For this family of systems, a well-suited solver was introduced in [16], generalizing the single solution criterion of [8] from zero-variate to univariate solution spaces.

In this paper, the problem of solving well-constrained ($n \times n$) piecewise polynomial systems is considered and, in particular, the main objective is focused on the reduction of the subdivision process. The solver proposed here follows [8] and extends the subdivision stage by two tests which are designed to speed up the elimination process of the no-root-containing domains. From n given constraints of the polynomial system, a *linear* blend (linear combination) is constructed in two different ways:

1. A maximum altitude blend (MAB), which maximizes the absolute value of the blending function in the midpoint of the domain.
2. A minimum slope blend (MSB), which minimizes the slope of the tangent hyperplane in the midpoint of the domain.

If the blending is *successful*, e.g. the blending function is strictly positive (negative) inside the domain, the domain is discarded from the subdivision process.

The rest of the paper is organized as follows. Section 2 briefly recalls the notions of a polynomial system, linear blend (linear combination), and the Bernstein–Bézier representation of a multivariate function. Section 3 introduces the maximum altitude blend (MAB) and minimum slope blend (MSB) and discusses their integration into the polynomial solver. Section 4 shows some examples where the solver may be applied. Finally, Section 5 identifies some possible future improvements and concludes.

2. Preliminaries

This paper deals with solving well-constrained (piecewise) polynomial systems and exploits Bernstein–Bézier's representation of multivariate functions and their linear blends. A brief survey of these topics will be given.

2.1. Bernstein's representation and the root-finding problem

Any univariate polynomial of degree d over domain $[\alpha, \beta] \subset \mathbb{R}$ can be expressed with respect to the Bernstein basis as

$$f(x) = \sum_{i=0}^{d} b_i B_i^d(x, \alpha, \beta), \tag{1}$$

where

$$B_i^d(x, \alpha, \beta) = \binom{d}{i} \frac{(x-\alpha)^i (\beta - x)^{d-i}}{(\beta-\alpha)^d} \tag{2}$$

is the i-th Bernstein polynomial over interval $[\alpha, \beta]$. Note that $B_i^d(x, \alpha, \beta),\ i = 0, \ldots, d$ form the basis of the $(d+1)$-dimensional linear space of polynomials of degree at most d on $[\alpha, \beta]$; see, e.g., [17]. Real numbers b_i are known as the *Bernstein–Bézier coefficients* of f.

The generalization to the multivariate case is rather straightforward, so any polynomial $f(\mathbf{x}) = f(x_1, \ldots, x_n)$ of degree d_i in variable x_i can be written as

$$f(\mathbf{x}) = \sum_{i_1=0}^{d_1} \cdots \sum_{i_n=0}^{d_n} b_{i_1 \ldots i_n} B_{i_1}^{d_1}(x_1, \alpha_1, \beta_1) \cdots B_{i_n}^{d_n}(x_n, \alpha_n, \beta_n), \tag{3}$$

and $B_{i_1}^{d_1}(x_1, \alpha_1, \beta_1) \cdots B_{i_n}^{d_n}(x_n, \alpha_n, \beta_n),\ i_1 = 0, \ldots, d_1, \ldots, i_n = 0, \ldots, d_n$ form the tensor product basis over domain $D = [\alpha_1, \beta_1] \times \cdots \times [\alpha_n, \beta_n]$.

Theorem 2.1 (*[17]*). *Let $(b_{i_1 \ldots i_n})_{0 \le i_1 \le d_1, \ldots, 0 \le i_n \le d_n}$ be the Bernstein coefficients of $f(\mathbf{x})$ on D. If $b_{i_1 \ldots i_n} > 0$ $(b_{i_1 \ldots i_n} < 0)$ for all feasible indices, then $f(\mathbf{x}) > 0$ $(f(\mathbf{x}) < 0)$ for all $\mathbf{x} \in D$.*

This *sufficient* condition of positivity (negativity) of a multivariate polynomial directly follows the convex hull property [17], and will be referred to as a *sign exclusion test* since only the signs of Bernstein coefficients need to be checked.

We consider the problem of finding all simultaneous roots of *well-constrained* (piecewise) polynomial systems over some D, which is formally stated as follows.

Definition 2.2. Consider the mapping $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^n$, such that each component $f_i,\ i = 1, \ldots, n$ of $\mathcal{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})]$ is a (piecewise) polynomial function in variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Then, every solution $\mathbf{x}$ of the system,

$$\mathcal{F}(\mathbf{x}) = \overrightarrow{0}, \tag{4}$$

is called a *root* of $\mathcal{F}$, and the set of all roots is known as the *zero set* of the mapping $\mathcal{F}$.

2.2. Linear blending of polynomial functions

Definition 2.3. Consider System (4). We call

$$a(\mathbf{x}) = a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \cdots + a_n f_n(\mathbf{x}), \tag{5}$$

$\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$, a *blend* of $\mathcal{F}$ and a *unit blend* if $\|\mathbf{a}\|_2 = 1$, where $\|.\|_2$ represents the Euclidean norm.

Remark 2.4. If no misunderstanding can occur, we refer to both the function $a(\mathbf{x})$ and the vector $(a_1, a_2, \ldots, a_n)$ as a blend.

Definition 2.5. We say that blend a is no-root contributing in domain D if either

$$a(\mathbf{x}) > 0, \quad \forall \mathbf{x} \in D, \tag{6}$$

or

$$a(\mathbf{x}) < 0, \quad \forall \mathbf{x} \in D. \tag{7}$$

Clearly, if there exists a no-root-contributing blend, system (4) has no real roots. By contradiction, if there is a root $\mathbf{r} \in D, f_i(\mathbf{r}) = 0$ for all $i = 1, \ldots, n$, then Eq. (5) gives $a(\mathbf{r}) = 0$, which violates the assumption that a is no-root contributing.

Example 2.6. For $n = 2$, system (4) is visualized as the intersection problem of two planar algebraic curves, over domain $D = [-1, 2] \times [-1, 2]$; see Fig. 1. A sequence of 15 different linear