Applied Mathematics and Computation 275 (2016) 195–212

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Particle-based meta-model for continuous breakpoint optimization in smooth local-support curve fitting

Akemi Gálvez[a], Andrés Iglesias[a,b,*]

[a] Department of Applied Mathematics and Computational Sciences, E.T.S.I. Caminos, Canales y Puertos, University of Cantabria, Avda. de los Castros, s/n, 39005 Santander, Spain
[b] Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, 274-8510 Funabashi, Japan

ARTICLE INFO

Keywords:
Particle swarm
Meta-model
Breakpoint optimization
Local-support curves
Curve fitting
CAD/CAM

ABSTRACT

This paper concerns the process of computing the underlying function of a given set of data points. In many cases, it is not possible to obtain an analytical solution for this problem so the goal is transformed into that of computing a meta-model instead. In this paper we seek to compute a smooth meta-model of such points based on local-support free-form parametric curves. Given an initial parameterization, our method applies a particle-based metaheuristic approach to determine optimal values for the breakpoints and poles of the fitting curve, which is well-known to be a continuous nonlinear optimization problem. The performance of our approach is evaluated by its application to two illustrative examples: a synthetic academic shape and a real-world shape. Our experimental results show that the proposed scheme performs very well, even for shapes with underlying functions exhibiting challenging features, such as self-intersections and sharp changes of curvature. Comparative results show that our approach outperforms previous approaches in terms of generality and fitting error accuracy.

1. Introduction

1.1. Motivation

Many industrial and engineering processes require performing parametric learning in order to obtain the underlying function of data. Very often, the mathematical structure of such a function cannot be readily obtained from data, so the problem is transformed into that of obtaining a *meta-model*, i.e., a model that mimics the behavior of the actual function but it is somehow computational cheaper to analyze and manipulate. A classical example arises in *Reverse Engineering*, where we seek to obtain a digitsal model of the surface of an already existing physical object [1,2]. A common approach in this field is to obtain a cloud of data points from the physical object by using a 3D laser scanner or other digitizing devices (coordinate measuring machines, CT scanners, light digitizers). This is a typical procedure in many applied and industrial fields, such as computer graphics, computer animation, virtual reality, computer-aided design and manufacturing (CAD/CAM), shoes industry, archeology (reconstruction of archeological assets), and in many other fields. Data can also be acquired from direct real measurements on a workpiece, as it typically happens in the construction of car bodies, ship hulls, airplane fuselage and other free-form objects [1,3–9]. Other

[*] Corresponding author at: Department of Applied Mathematics and Computational Sciences, E.T.S.I. Caminos, Canales y Puertos, University of Cantabria, Avda. de los Castros, s/n, 39005 Santander, Spain. Tel.: +34 942202062; fax: +34 942201703.
E-mail address: iglesias@unican.es (A. Iglesias).
URL: http://personales.unican.es/iglesias (A. Iglesias)

http://dx.doi.org/10.1016/j.amc.2015.11.050
0096-3003/© 2015 Elsevier Inc. All rights reserved.

example comes from medical imaging, where inner organs of our body are analyzed through a series of images acquired by computer tomography or magnetic resonance. Then, digital geometry processing is applied to generate a fully 3D volumetric image of the organ for diagnostic and therapeutic purposes. A similar approach is also taken in many areas of industry for internal inspection and analysis of physical components of a workpiece.

A common factor of all these instances is their huge number of data, typically ranging from thousands to millions of data points assumed to lie on an unknown curve or surface. Unfortunately, in most cases we have no information about these geometric entities beyond their data points. As a consequence, it is very difficult to figure out the mathematical structure of such curves or surfaces and we are forced to replace them by meta-models via data fitting according to some prescribed criteria. Of course, such meta-models must be precise enough to represent the available information with high accuracy so that the user can perform typical engineering operations such as design optimization, system simulation, instrumentation and measurement, control engineering, quality assessment, life testing, and many others. In general, two approaches are usually taken: interpolation (where the curve/surface is constrained to pass through all input data points) and approximation (where the curve/surface is expected to pass near the data points). In industrial applications, approximation is preferred over interpolation because typically real-world data are irregularly sampled and strongly affected by noise, making interpolation both impractical and unnecessary. This is also the approach taken in this paper. In particular, we focus on the case of data fitting with parametric curves, as described in next section.

1.2. Meta-model representation

Approximation meta-models for data fitting with curves can be classified into three groups regarding its mathematical structure: explicit, implicit and parametric. In explicit representation, the dependent variables are “explicitly” expressed as functions of the independent variables, with each type of variables placed on different sides of the equation. For instance, explicit planar curves are given by the expression: $y = f(x)$. On the contrary, in implicit representation we have a function containing both variables in the form $F(x, y) = 0$. The curve is then formed by all pairs (x, y) satisfying this implicit relationship. This representation is more general than the explicit one: all explicit functions can be transformed into implicit ones (just take $F(x, y) = y - f(x)$ in our example) but the opposite is not always true. In particular, an implicit function can represent curves with multiple branches, self-intersections or loops, while an explicit function cannot.

The parametric representation is the most common in commercial software and industrial fields. In that representation, each coordinate variable is independently represented as a function of a parameter (say t) such as $\{x = f(t),\ y = g(t)\}$ for planar curves. This representation allows a quick computation of the coordinates of all points on a curve. It is also useful to define a curve segment constraining the parameter to intervals [10]. This is an interesting feature because curves are usually bounded in real-world settings. Finally, parametric curves are very flexible, since variables x and y are fully independent of each other. As a result, they are able to represent a broad variety of shapes. This feature is particularly noticeable for the so-called *free-form parametric functions*, such as Bézier, B-splines and NURBS. Free-form curves are based on *poles* (also called *control points*) which are connected by linear segments to form a control polygon that roughly determine the shape of the curve. Instead of being described by polynomial equations, free-form curves are typically encoded in terms of their poles, and the degree (alternatively, the order) of the curve. Generally, this description leads to a polynomial equation obtained as a linear combination of basis functions with the coefficients being the poles.

Free-form curves can be of two types. First type corresponds to *global-support* curves, when the basis functions are defined over the whole curve domain. As a result, they exhibit global control: moving a pole modifies the entire curve. Second type corresponds to *local-support* curves: their basis functions are defined on a subset (usually, a bounded interval) of the whole domain, so they exhibit local control: only a portion of the curve is affected by pole movement. Since local control is better suited for interactive design and manipulation, local-support approaches (driven by piecewise functions, such as B-splines and NURBS) have become prevalent in CAD/CAM and computer graphics. In this paper we focus on smooth local-support free-form curves, particularly B-splines.

In spite of their popularity and their broad range of applications, using B-splines is still challenging because they depend on many different continuous variables (data parameters, breakpoints, and poles) in a difficult nonlinear and interrelated way [11–15]. For instance, the computation of breakpoints requires a previous parameterization which, at its turn, requires previous breakpoint determination, and so on [16,17]. Mathematically, this implies that the fitting problem cannot be partitioned into independent sub-problems for the different sets of variables [18–21]. A classical way to overcome this limitation is to make a choice of values for some of those variables and compute the values of all the others. Classical choices are uniform, chordal, and centripetal methods for data parameterization [17]. Then, breakpoints can be computed by using either the uniform or the averaging methods. But these methods are far from being optimal in many situations, as they do not fully reflect the structure of data [22]. Furthermore, in general there is no analytical solution to the problem of computing the breakpoints from a given parameterization and the data points [14,23] and, therefore, we must rely on numerical procedures instead. The typical approach is to formulate the problem as a continuous nonlinear optimization problem [22,24]. Unfortunately, the traditional numerical optimization techniques are not enough to solve this problem. Among the alternatives suggested in the literature, those based on artificial intelligence techniques are receiving increasing attention during the last few years (see our discussion in Section 2). However, the solutions reported so far are partial and applicable only to some particular cases. More recently, the scientists and engineers have turned their attention to bio-inspired computation, a field where the interplay between nature and computers has led to improved schemes to solve many problems in mathematics and computer science, including difficult optimization