Boundary geometry fitting with Bézier curves in PIES based on automatic differentiation

Krzysztof Szerszeń [0000-0001-9256-2622] and Eugeniusz Zieniuk [0000-0002-6395-5096]

University of Bialystok, Institute of Computer Science
15-245 Białystok, Konstantego Ciołkowskiego 1M, Poland
{kszerszen, ezieniuk}@ii.uwb.edu.pl

Abstract. This paper presents an algorithm for fitting the boundary geometry with Bézier curves in the parametric integral equation system (PIES). The algorithm determines the coordinates of control points by minimizing the distance between the constructed curves and contour points on the boundary. The minimization is done with the Adam optimizer that uses the gradient of the objective function calculated by automatic differentiation (AD). Automatic differentiation eliminates error-prone manual routines to evaluate symbolic derivatives. The algorithm automatically adjusts to the actual number of curves and their degrees. The presented tests show high accuracy and scalability of the proposed approach. Finally, we demonstrate that the resulting boundary may be directly used by the parametric integral equation system (PIES) to solve the boundary value problem in 2D governed by the Laplace equation.

Keywords: automatic differentiation, parametric curve fitting, Bézier curves, parametric integral equation system (PIES), boundary value problems

1 Introduction

One of the main difficulties during computer simulation of boundary value problems (BVP) is the appropriate definition of the computational domain. Typically it is done by dividing the problem domain into finite elements (FEM) [1] or only the boundary of that domain into boundary elements (BEM) [2]. However, such discretization is extremely laborious as it requires hundreds or thousands of elements and even more nodes are necessary to declare them. The alternative is to introduce the mathematical and geometric tools used in computer graphics and CAD/CAM systems. This is used in the parametric integral equation system (PIES) where the boundary of the computational domain is bounded by parametric curves [3] and surfaces [4]. Moreover, due to the analytical integration of the boundary geometry directly in the PIES formula, there is a lot of freedom in choosing the appropriate structure of the boundary representation. Our previous studies have shown that it is particularly effective to declare such boundary geometries by employing Bézier parametric curves defined by a small set of control points. This allows for a continuous representation of the boundary and the number of the curves is significantly smaller than finite or boundary elements required to solve the same problem.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39

Despite these advantages, parametric curves also have some implementation difficulties since its shape is determined by control points that generally do not lie on the curve. Curve fitting is a fundamental problem in computer graphics and has become a very active scientific area. Mathematically, it can be formulated as optimization problem to minimize the distance between the points describing the approximated shape and the points on the parametric curve. There is a rich literature to solve such optimization problem with several linear [5] and non-linear [6] least-squares techniques. Another approaches are based on biologically inspired solutions: genetic algorithms [7], simulated annealing [8], particle swarm optimization [9], evolutionary algorithms [10], artificial immune systems [11]. In [12] neural network-based curve fitting technique is presented. However, most existing algorithms are based on gradient descent minimization [13]. On the other hand, analytical evaluation of gradients is a labor-intensive task and sensitive to human errors, especially in the case of real problems defined by many design variables related to the coordinates of control points.

This paper attempts a new look at the parametric curve fitting applied to the boundary approximation in PIES. This is done by minimizing the distance between the constructed curves and the contour points on the boundary with the Adam optimizer [14], where the required derivatives of the objective function are computed by automatic differentiation (AD). The idea of AD is based on a decomposition of an input function into elementary operations, whose local derivatives are easy to compute. While AD has a long history and is supported by many publications [15,16], the recent revolution in deep learning and artificial intelligence has brought a new stage in the development of advanced AD tools and new optimization algorithms. Popular libraries such as TensorFlow [17] and PyTorch [18] provide efficient AD and optimization algorithms for large models with thousands or even millions of parameters. The results of the presented tests illustrate good fitting properties of the proposed algorithm for various shapes of the boundary and high scalability. The fitted boundaries are directly used in PIES to solve BVP governed by the Laplace equation.

2 Defining the boundary by Bézier curves in PIES

We consider a boundary value problem governed by the Laplace equation defined in a domain Ω with a boundary Γ, as shown in Fig. 1a.

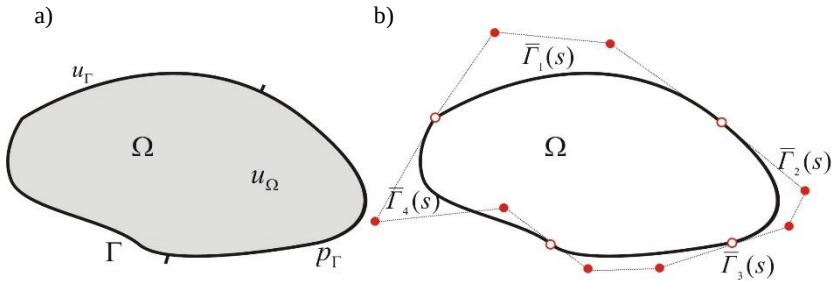

Fig. 1. BVP defined in the domain Ω with the boundary Γ (a), boundary description in PIES with 4 Bézier curves and 12 control points (b).

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39

The boundary of that domain is described in PIES by a set of Bézier curves. A single Bézier curve of degree m is defined by the location of the $m+1$ control points P_i and is mathematically described as

$$\overline{\Gamma}(s) = \sum_{i=0}^{m} \binom{m}{i} s^i (1-s)^{m-i} P_i , \tag{1}$$

where s is a parametric coordinate along the curve $(0 \le s \le 1)$. Fig. 1b shows a practical definition of the boundary formed by joining 4 cubic Bézier curves. The formula of PIES for 2D BVP governed by the Laplace equation is presented below

$$0.5u_l(\bar{s}) = \sum_{j=1}^{n} \int_{s_{j-1}}^{s_j} \left\{ \overline{U}_{lj}^{\,*}(\bar{s},s) p_j(s) - \overline{P}_{lj}^{\,*}(\bar{s},s) u_j(s) \right\} J_j(s) ds, \tag{2}$$

where $s_{j-1} \le s \le s_j$, $s_{l-1} \le \bar{s} \le s_l$, $s_{j-1} \le s \le s_j$, $l = 1,2,...,n$.

The Bézier curves $\overline{\Gamma}(s)$ are included analytically in the kernels $\overline{U}_{lj}^{\,*}(\bar{s},s)$ and $\overline{P}^{*}_{lj}(\bar{s},s)$ written as

$$\overline{U}_{lj}^{\,*}(\bar{s},s) = \ln \frac{1}{[\eta_1^2 + \eta_2^2]^{0.5}}, \tag{3}$$

$$\overline{P}_{lj}^{\,*}(\bar{s},s) = \frac{\eta_1 n_1^{(j)}(s) + \eta_2 n_2^{(j)}(s)}{\eta_1^2 + \eta_2^2}, \tag{4}$$

where $$\eta_1 = \overline{\Gamma}_l^{(1)}(\bar{s}) - \overline{\Gamma}_j^{(1)}(s), \ \eta_2 = \overline{\Gamma}_l^{(2)}(\bar{s}) - \overline{\Gamma}_j^{(2)}(s). \tag{5}$$

Moreover, $n_1^{(j)}(s), n_2^{(j)}(s)$ denote the components of the normal vector to the boundary and $J_j(s)$ is the Jacobian. The reader can find details of the PIES mathematical formulation in several previous papers, for example in [3].

3 Proposed algorithm

Let G_Γ be a set of data points sampled along the boundary and $G_{\overline{\Gamma}}$ is a set of points lying on Bézier curves. The objective function $loss_{L2}(G_\Gamma, G_{\overline{\Gamma}})$ of our optimization problem is the L_2 distance between G_Γ and $G_{\overline{\Gamma}}$. The gradient of the objective function with respect to the corresponding control points P of the curves is written as

$$\frac{\partial loss(P)}{\partial P} = \frac{\partial loss_{L2}(G_\Gamma, G_{\overline{\Gamma}})}{\partial \overline{\Gamma}} \frac{\partial \overline{\Gamma}}{\partial P}. \tag{6}$$

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39

The gradient (6) can be derived analytically, however, is laborious and prone to human errors. Therefore, we compute it employing AD which decomposes the derivative of the objective function into those of elementary operations based on the chain rule. This decomposition can be described by a computational graph that shows the relations between individual differentials. For demonstrating the procedure, an elementary case with one cubic Bézier curve defined by 4 control points is analyzed below. In this case, the formula (6) reduces to the following form

$$\frac{\partial loss(P)}{\partial P} = \frac{\partial}{\partial P}\left(G_\Gamma - CAP\right)^2, \tag{7}$$

where C,A,P are referred to a matrix representation for the cubic Bézier curve

$$C = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix},\ A = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},\ P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}. \tag{8}$$

Finally, the computational graph for formula (7) is presented below.

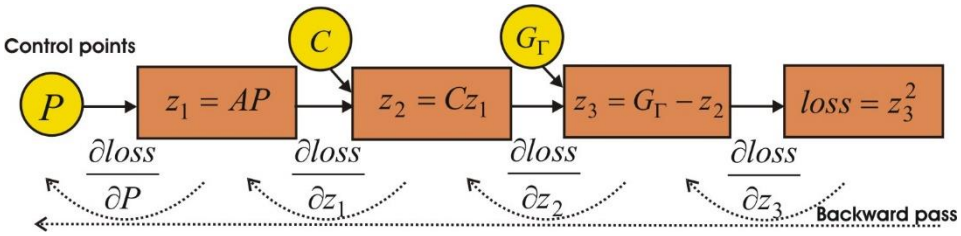

Fig. 2. Structure of the computational graph for a single cubic Bézier curve.

Tab. 1 contains the vertices z_i corresponding to the intermediate computed variables for elemental operations and their derivatives for forward and backward data flow.

Table 1. Forward and backward propagation of derived values for the graph shown in Fig. 2.

	Forward pass	**Backward pass**
$z_1 = AP$	$\frac{\partial z_1}{\partial P} = A$	$\frac{\partial loss}{\partial P} = \frac{\partial loss}{\partial z_1}\frac{\partial z_1}{\partial P} = -2CA$
$z_2 = Cz_1$	$\frac{\partial z_2}{\partial P} = \frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial P} = CA$	$\frac{\partial loss}{\partial z_1} = \frac{\partial loss}{\partial z_2}\frac{\partial z_2}{\partial z_1} = -2C$
$z_3 = G_\Gamma - z_2$	$\frac{\partial z_3}{\partial P} = \frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial P} = -CA$	$\frac{\partial loss}{\partial z_2} = \frac{\partial loss}{\partial z_3}\frac{\partial z_3}{\partial z_2} = -2$
$loss = z_3^2$	$\frac{\partial loss}{\partial P} = \frac{\partial loss}{\partial z_3}\frac{\partial z_3}{\partial P} = -2CA$	$\frac{\partial loss}{\partial z_3} = 2$

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39

The computational graph can dynamically update its structure for more Bézier curves and control points. The full diagram of the procedure with backward mode AD as best suited for our problem is shown in Fig. 3.

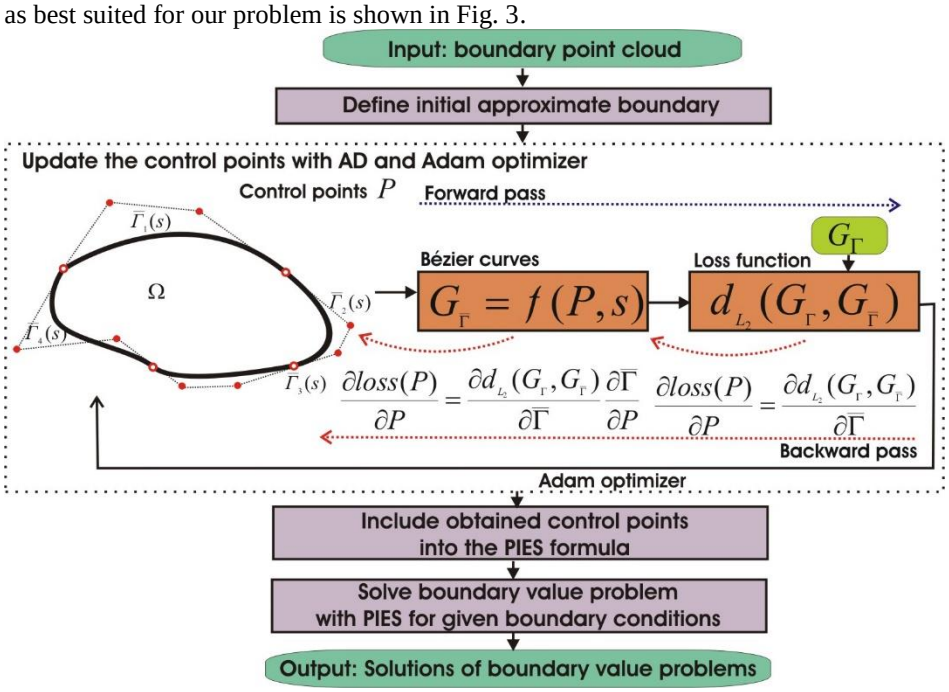

Fig. 3. Schematic flowchart of the proposed curve fitting combined with PIES.

To determine the coordinates of control points, the ADAM [14] optimization algorithm is used characterized by fast convergence, and resistance to local minima.

4 Results and evaluation

The scheme given in Fig. 3 is implemented in the PyTorch framework providing access to AD and Adam optimizer modules. Fig. 4 shows the results of our experimental evaluation for 4 identified boundaries generated from 16 to 40 cubic Bézier curves.

Table 2. Convergence of the objective functions during iterations and the PIES accuracy.

$loss_{L2}(G_\Gamma, G_{\bar{\Gamma}})$

Iteration	**A**	**B**	**C**	**D**
1	75.3377	186.8021	212.6234	195.2782
50	25.1174	125.8349	147.8667	98.5266
100	1.0304	81.2112	87.9212	20.3172
200	0.0935	17.3265	1.5679	0.1621
400	0.0685	0.0831	0.0657	0.0954
PIES (L_2 error norm)				
400	0.1567	0.1235	0.3643	0.3025

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39

The curves require 48 and 120 control points to be defined. The data set G_Γ consists of evenly sampled points along the boundary. The first three shapes are sampled into 160 points and the last one into 400 points.

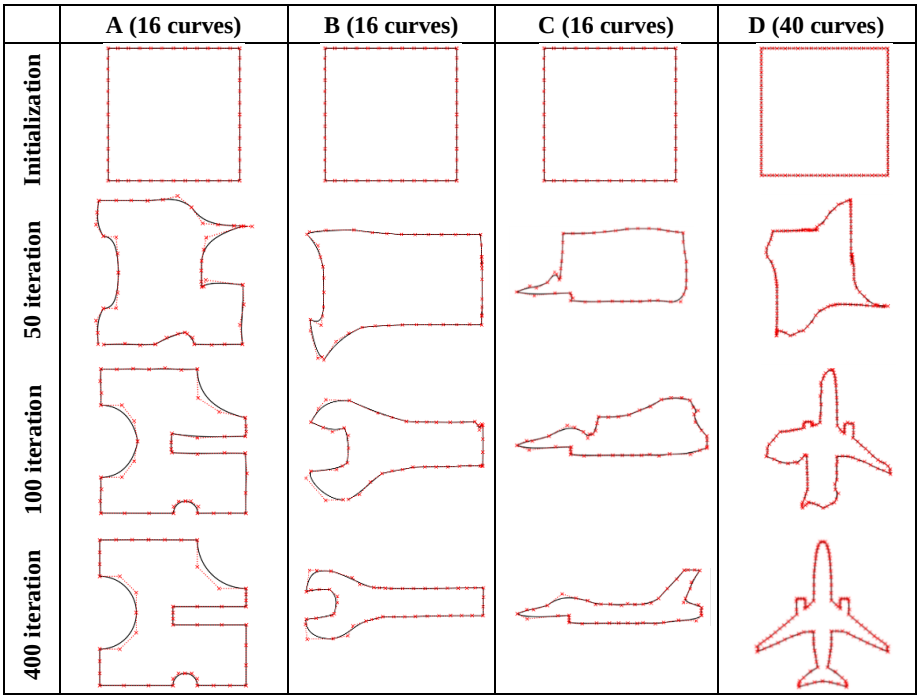

Fig. 4. Iterations 1, 50, 100, and 400 in the fitting process.

As it can be seen from the graphic results, all tested shapes are fitted successfully. The related numerical results are listed in Tab. 2. Finally, the final boundaries are directly used in PIES to simulate BVP governed by the Laplace equation for the following Dirichlet boundary conditions

$$\phi(x_1, x_2) = \cos(x_1)\cosh(x_2) + \sin(x_1)\sinh(x_1)\,. \tag{9}$$

This is possible since the control points are included analytically in the kernels (3,4). In order to solve the problem on the boundary, the collocation method [3] is adopted with 6 colocation points per each Bézier curve. The last row in Tab. 2 shows the L_2 error norm of the PIES solutions for all shapes from the 400th iteration of the fitting.

5 Conclusions

The results show that the proposed approach can be applied not only to academic but also to real-life problems with hundreds of design variables considered as the positions of control points. The boundary defined by the Bézier curves is integrated analytically

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39

with the PIES computational method so that it can be directly used to solve BVP, as shown in the example. The approach can use different types of objective functions and optimization algorithms. We also hope to extend the research to study the boundary reconstruction from an unstructured point cloud as well to 3D problems.

References

1. Zienkiewicz, O.C., Taylor R.L.: The finite element method for solid and structural mechanics. Elsevier (2005).
2. Brebbia, C.A., Telles, J.C.F., Wrobel, L.C.: Boundary element techniques: theory and applications in engineering. Springer Science & Business Media (2012).
3. Zieniuk, E., Szerszeń, K.: NURBS curves in direct definition of the shapes of the boundary for 2D Stokes flow problems in modified classical BIE. Applied Numerical Mathematics 132, 111-126 (2018).
4. Zieniuk, E., Szerszeń, K.: Nonelement boundary representation with Bézier surface patches for 3D linear elasticity problems in parametric integral equation system (PIES) and its solving using Lagrange polynomials. Numerical Methods for Partial Differential Equations 34(1), 51-79 (2018).
5. Piegl, L., Tiller, W.: The NURBS book. Springer Science & Business Media (1996).
6. Borges, C.F., Pastva, T.: Total least squares fitting of Bézier and B-spline curves to ordered data. Computer Aided Geometric Design 19(4), 275-289 (2002).
7. Zhou, M., Wang, G.: Genetic algorithm-based least square fitting of B-Spline and Bézier curves. Journal of Computer Research and Development 42(1), 134–143 (2005).
8. Zhang, J., Wang, H.: B-Spline curve fitting based on genetic algorithms and the simulated annealing algorithm. Computer Engineering & Science 33(3), 191–193 (2011).
9. Gálvez, A., Iglesias, A.: Efficient particle swarm optimization approach for data fitting with free knot B-splines. Computer-Aided Design 43(12), 1683-1692 (2011).
10. Pandunata, P., Shamsuddin, S.M.H.: Differential evolution optimization for Bézier curve fitting. In 2010 Seventh International Conference on Computer Graphics, Imaging and Visualization, 68-72 (2010).
11. Iglesias, A., Gálvez, A., Avila, A.: Discrete Bézier curve fitting with artificial immune systems. Studies in Computational Intelligence 441, 59-75 (2013).
12. Bishop, C.M., Roach, C.M.: Fast curve fitting using neural networks. Review of scientific instruments 63(10), 4450-4456 (1992).
13. Samir C., Absil, P.A., Srivastava, A., Klassen, E.: A gradient-descent method for curve fitting on Riemannian manifolds. Foundations of Computational Mathematics 12(1), 49-73 (2012).
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
15. Margossian, C.C.: A review of automatic differentiation and its efficient implementation. Wiley interdisciplinary reviews: data mining and knowledge discovery 9, 1305e (2019).
16. Neidinger, R.D.: Introduction to automatic differentiation and MATLAB object-oriented programming. SIAM review, 52(3), 545-563 (2010).
17. Abadi, M. et al.: {TensorFlow}: A System for {Large-Scale} Machine Learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), 265-283 (2016).
18. Paszke, A. et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32 (2019).

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_39